**Lab Assignment#1**

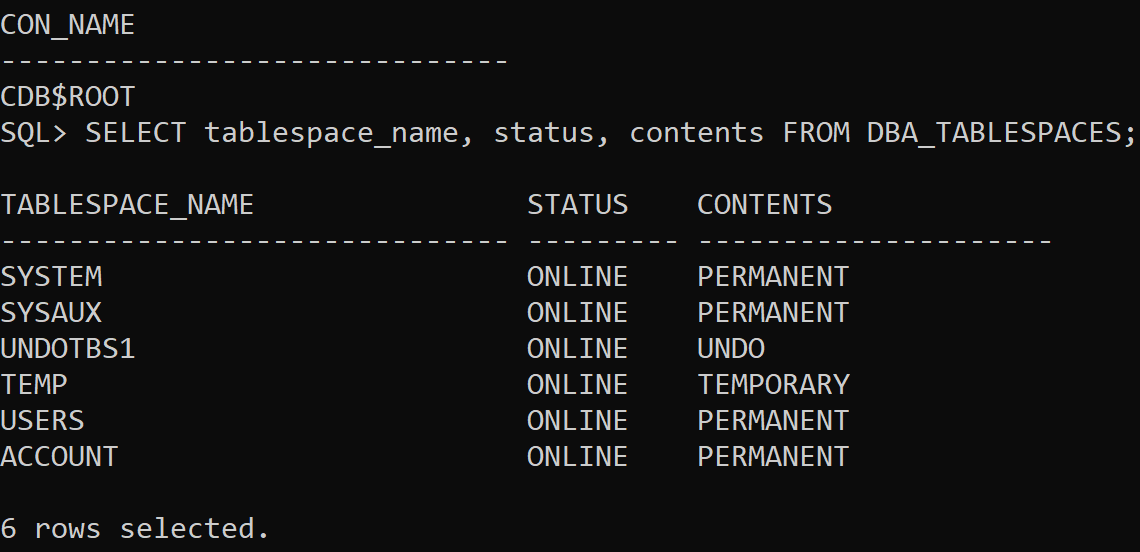
1. **Locate the tablespace, data files, control files, online redo log files and parameter file (spfile) in your database using different dynamic performance views.**

a).**Tablespaces:**

To get information about tablespaces, we can query the DBA\_TABLESPACES view.

**SYNTAX**: SELECT tablespace\_name, status, contents FROM DBA\_TABLESPACES;

**OUTPUT:**



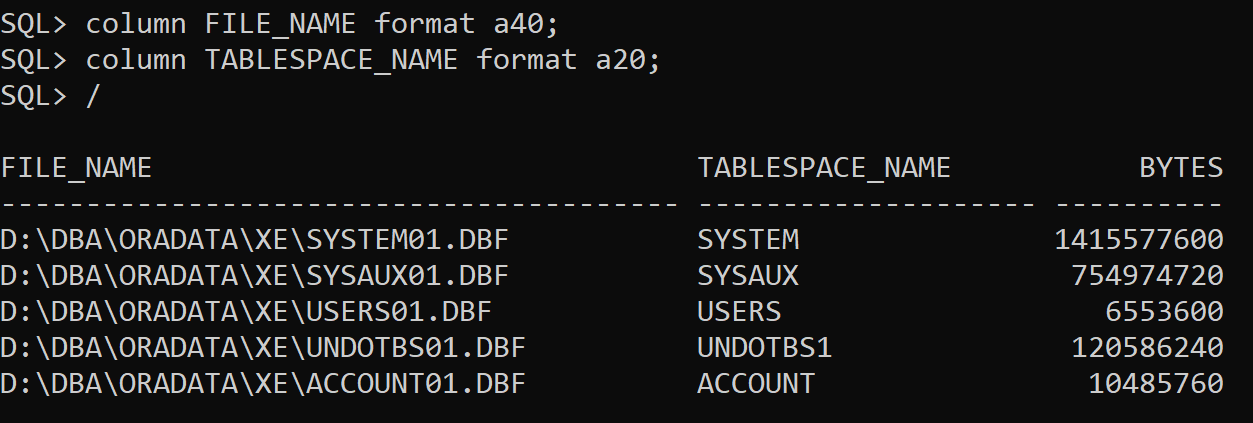
b).**Data files:**

To retrieve information about data files, we can use the DBA\_DATA\_FILES view.

**SYNTAX**: SELECT file\_name, tablespace\_name, bytes FROM DBA\_DATA\_FILES;

**OUTPUT:**

****

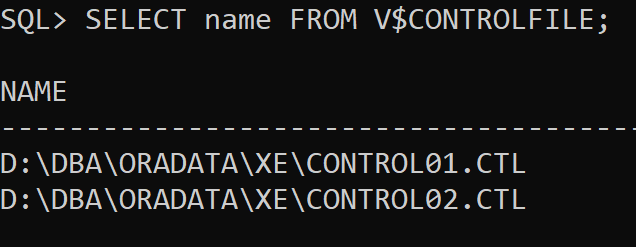
****

c).**Control files:**

The control file information can be obtained from the V$CONTROLFILE view.

**SYNTAX**: SELECT name FROM V$CONTROLFILE;

**OUTPUT:**

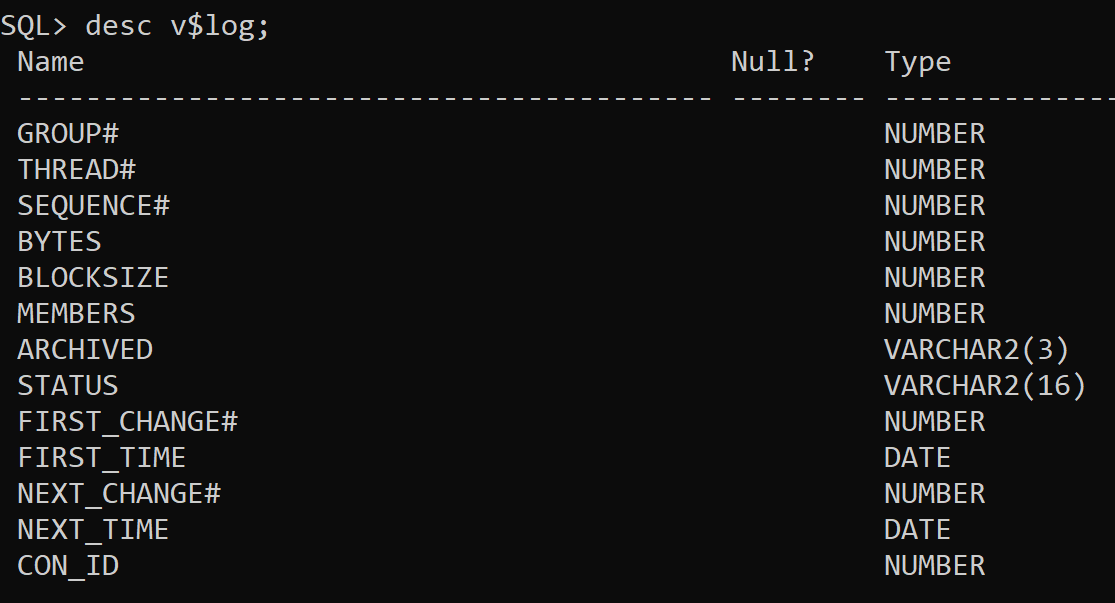
****

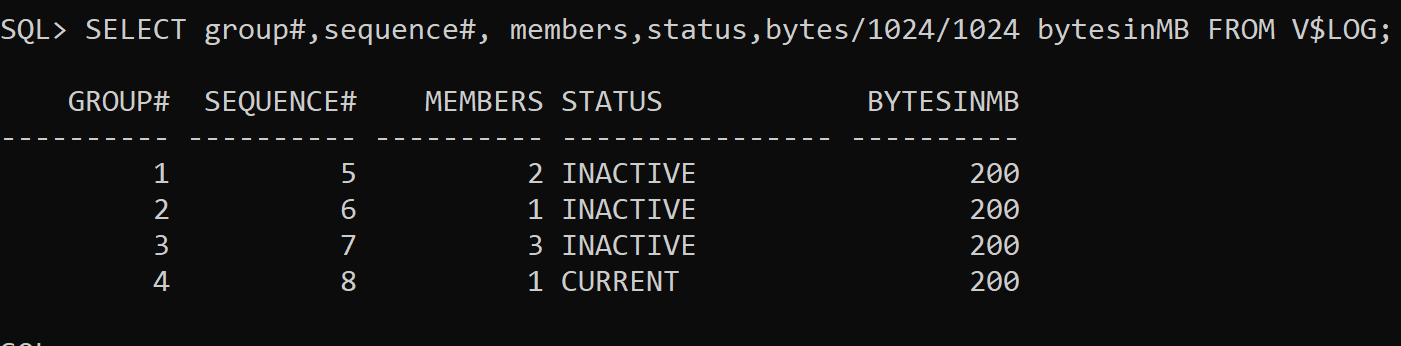
d).**Online Redo log files:**

To get information about online redo log files, we can use the ‘V$LOG’ view.

**SYNTAX**: SELECT group#, sequence# ,members, status, bytes/1024/1024 bytesinMB FROM V$LOG;

**OUTPUT:**

****

****

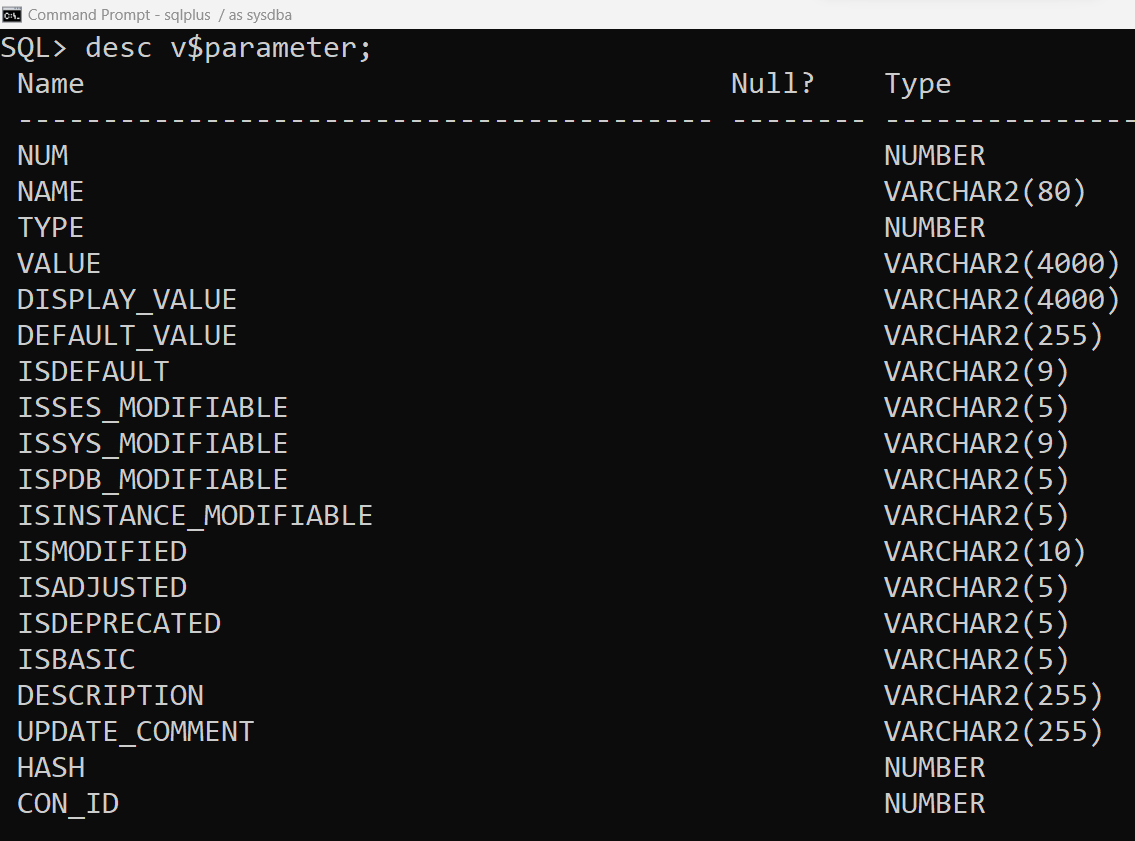
e).**Parameter files(spfile):**

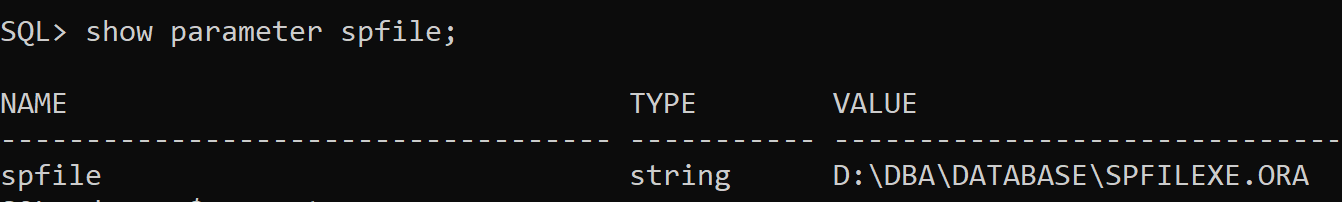
To find the location of the server parameter file (SPFILE), we can query the V$SPPARAMETER view.

**SYNTAX**: desc v$parameter;

show parameter spfile;

**OUTPUT:**

****

****

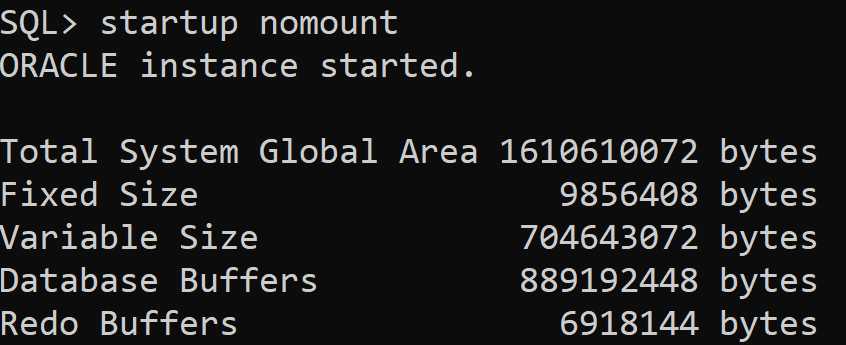
1. **Practice different startup and shutdown modes.**

**Database Startup Modes**

* **NOMOUNT Mode:** It reads only SPFILE or PFILE and allocating SGA.

**Syntax:** startup nomount

**Output:**



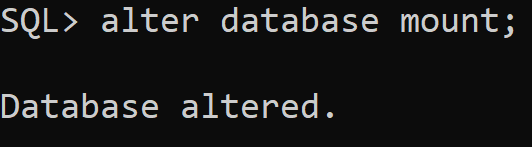
* **MOUNT Mode:** It reads SPFILE or PFILE and open control files.

**Syntax:** startup mount

Or

alter database mount;

**Output:**



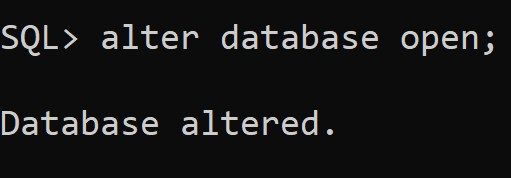
* **OPEN Mode:** It reads SPFILE or PFILE , open control files, datafiles and redo log files.

**Syntax:** startup

Or

alter database open;

**Output:**



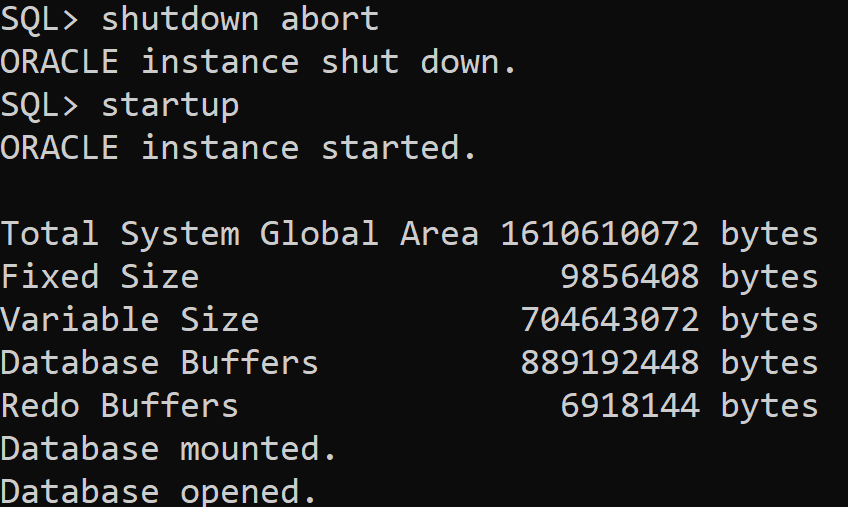
**Database Shutdown Modes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Shutdown Modes** | **Abort Modes** | **Immediate modes** | **Transactional Modes** | **Normal Modes** |
| Allows new connections | NO | NO | NO | NO |
| Waits until current sessions end | NO | NO | NO | YES |
| Waits until current transactions end | NO | NO | YES | YES |
| Forces a checkpoint and closes files | NO | YES | YES | YES |

* **ABORT Mode:**

**Syntax:** shutdown abort

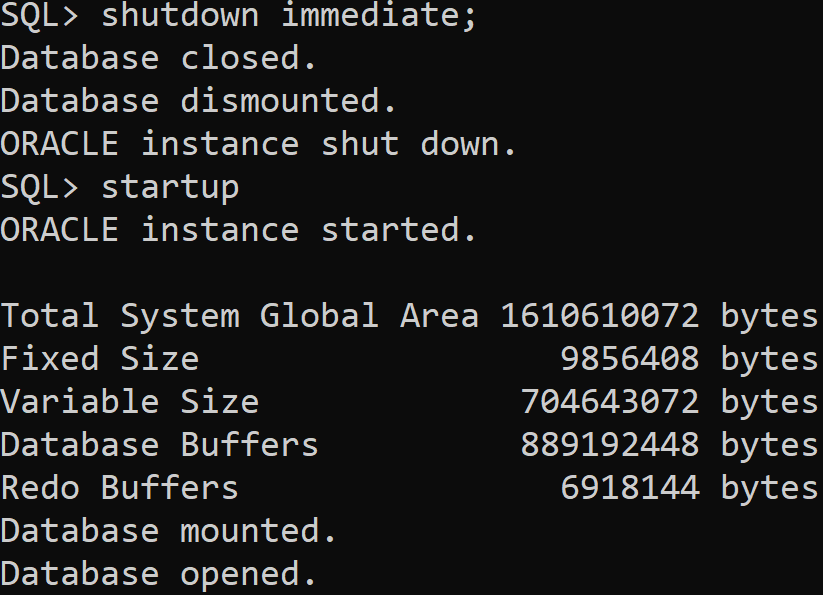
**Output:**

****

* **IMMEDIATE Mode:**

**Syntax:** shutdown immediate;

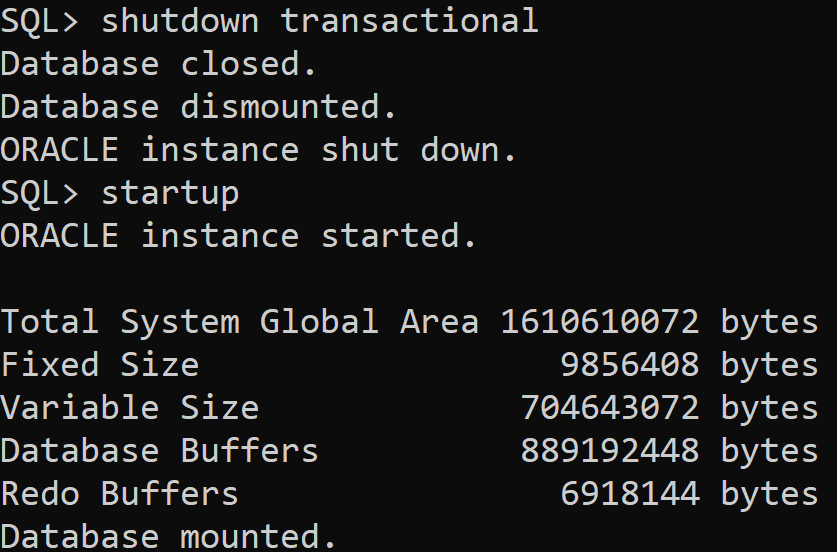
**Output:**

****

* **TRANSACTIONAL Mode:**

**Syntax:** shutdown transactional

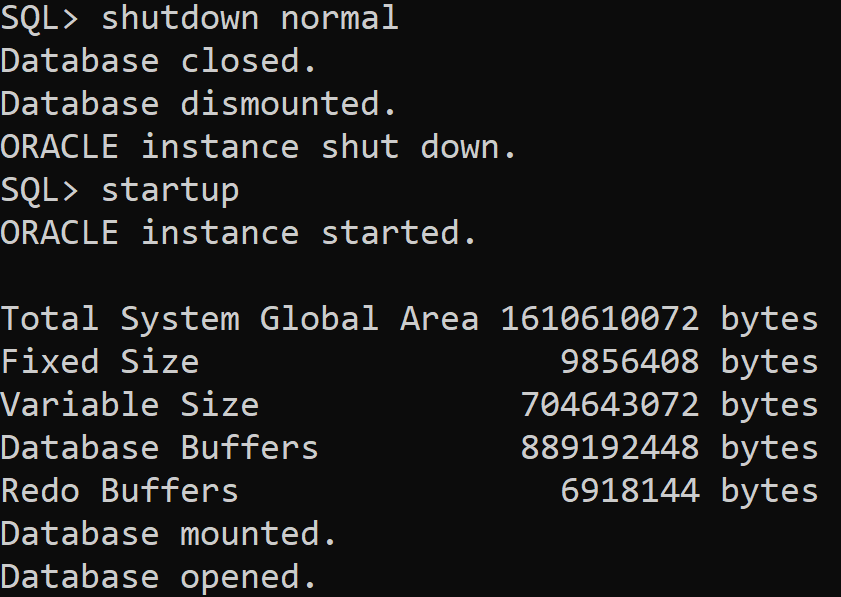
**Output:**

****

* **NORMAL Mode:**

**Syntax:** shutdown normal

**Output:**

****

1. **User Management**
   1. **User creation**
   2. Create three new database users named user1, user2, and user3
   3. Each new user is identified by the password 12345.
   4. Their default tablespace is USERS.
   5. Their temporary tablespace is TEMP.
   6. The quota is 20K on USERS.
   7. The profile is the Oracle database default profile.

**Syntax:**

SQL> create user user1 identified by 12345

2 default tablespace users

3 temporary tablespace temp

4 quota 20k on users

5 profile default;

SQL> create user user2 identified by 12345

2 default tablespace users

3 temporary tablespace temp

4 quota 20k on users

5 profile default;

SQL> create user user3 identified by 12345

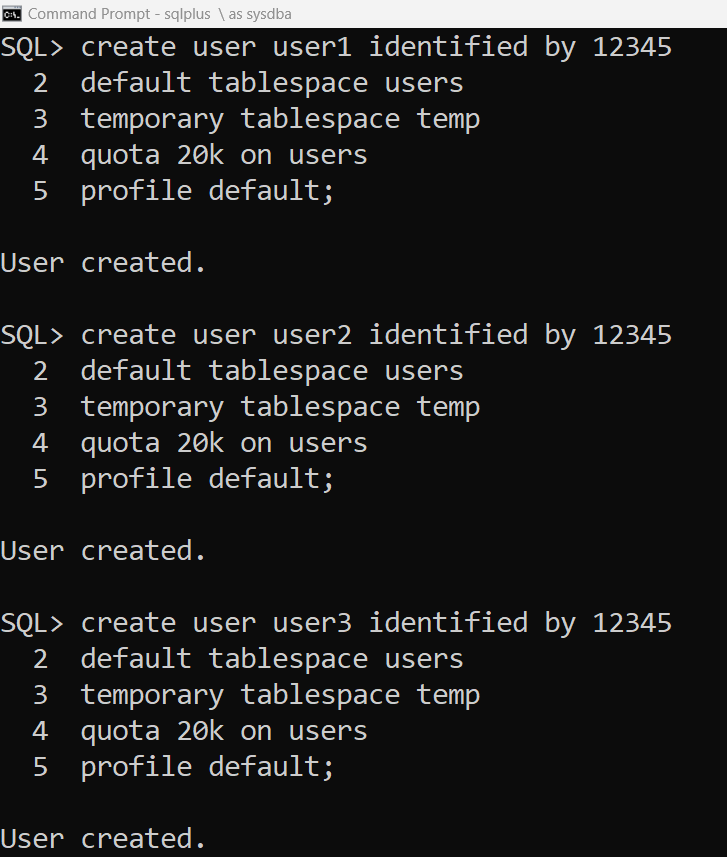
2 default tablespace users

3 temporary tablespace temp

4 quota 20k on users

5 profile default;

**Output:**

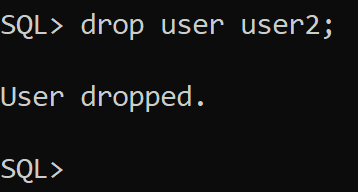
****

* 1. Delete the user with and without having object

Delete a user with having objects:

**Syntax:** drop user user2;

**Output:**

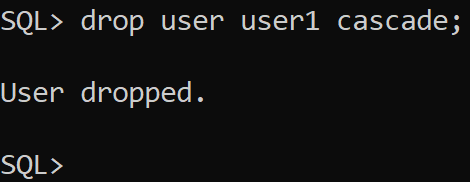


Delete a user without having objects:

**Syntax:** drop user user1 cascade;

-- Cascade deletes all objects owned by the user.

**Output:**



* 1. **Granting Privileges**
     1. Mention different system and object privileges

**System Privileges:**

CREATE SESSION: Allows a user to log in to the database.

DBA: Provides all database administrative privileges.

RESOURCE: Provides basic resource privileges for creating tables, sequences, procedures, etc.

SELECT ANY TABLE: Allows the user to select data from any table.

**Object Privileges:**

SELECT: Allows the user to query data from a specified table or view.

INSERT: Allows the user to insert rows into a specified table.

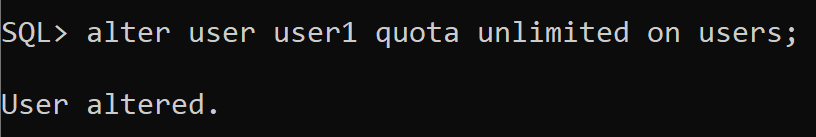
UPDATE: Allows the user to update rows in a specified table.

DELETE: Allows the user to delete rows from a specified table.

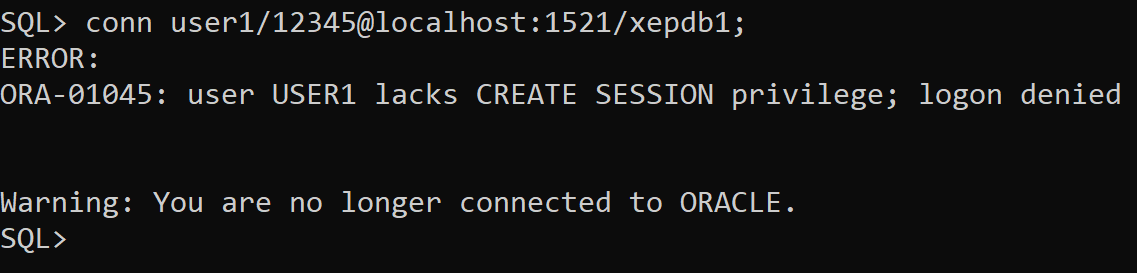
* + 1. Alter the user named **User1** to assign an unlimited quota on the **USERS** tablespace.

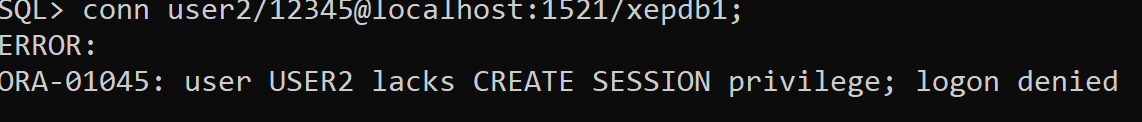
**Syntax:** alter user user1 quota unlimited on users;

**Output:**

****

* + 1. Attempt to connect to the database as user **User1**, and **User2**, in turn.  Why can’t you connect?  Grant **User1** and **User2** the privilege needed to connect to the system.

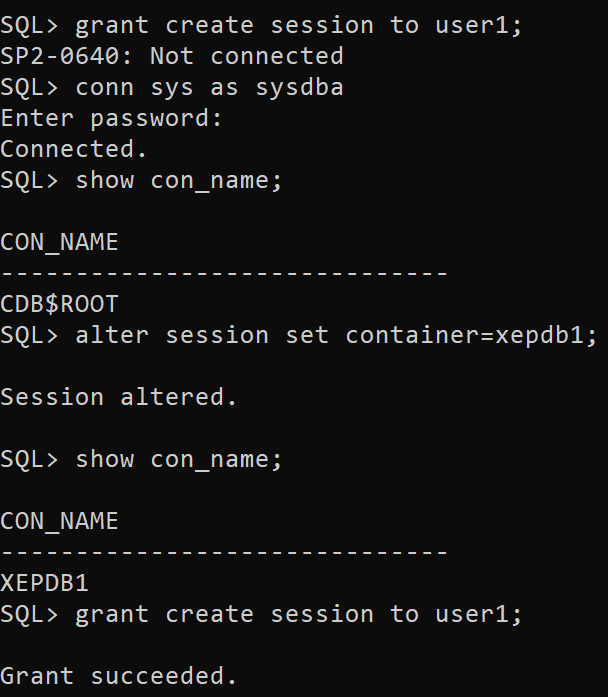




* ***Cannot connect to User1 and User2 because they are not granted to have privilege.***

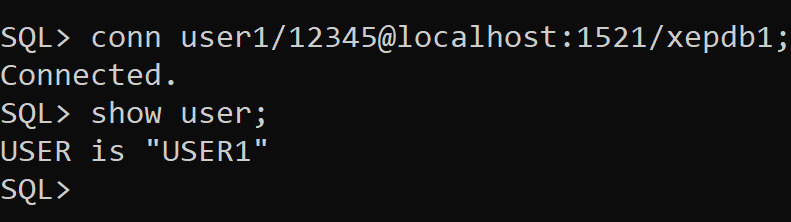
**Syntax:** grant create session to user1;

**Output:**

****

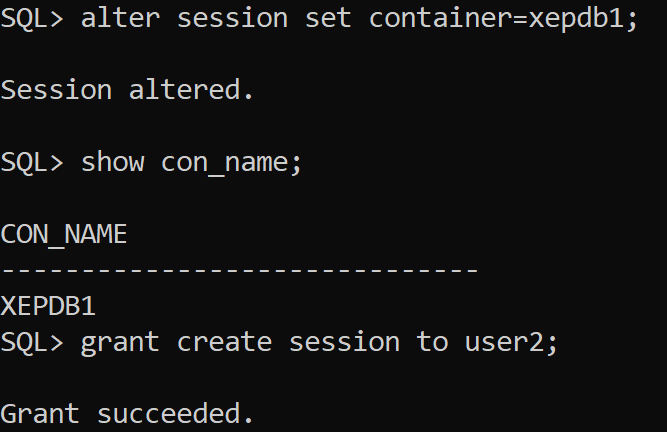
**Syntax:** conn user1/12345@localhost:1521/xepdb1;

**Output:**

****

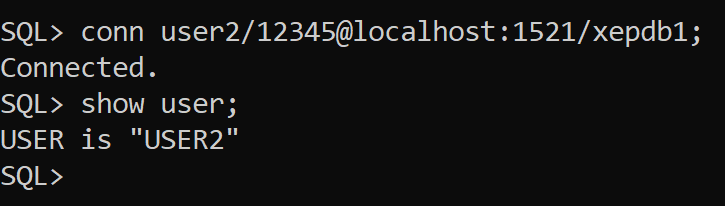
**Syntax:** grant create session to user2;

**Output:**

****

**Syntax:** conn user2/12345@localhost:1521/xepdb1;

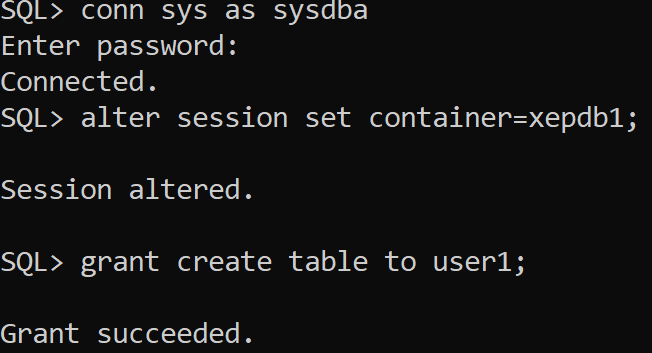
**Output:**

****

* + 1. Creating a table by a user – insert row data andquery the table to list the rows you inserted to verify their existence.

**Syntax:** grant create table to user1;

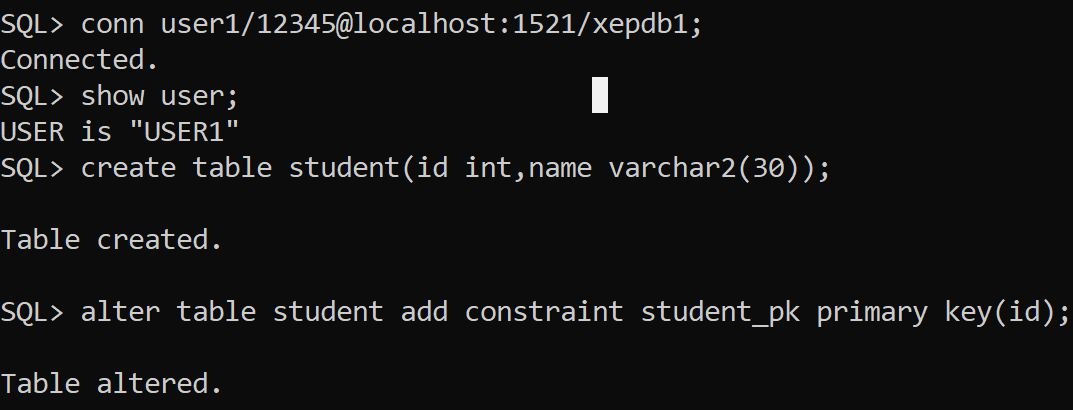
**Output:**



**Syntax:**  create table student(id int , name varchar2(30));

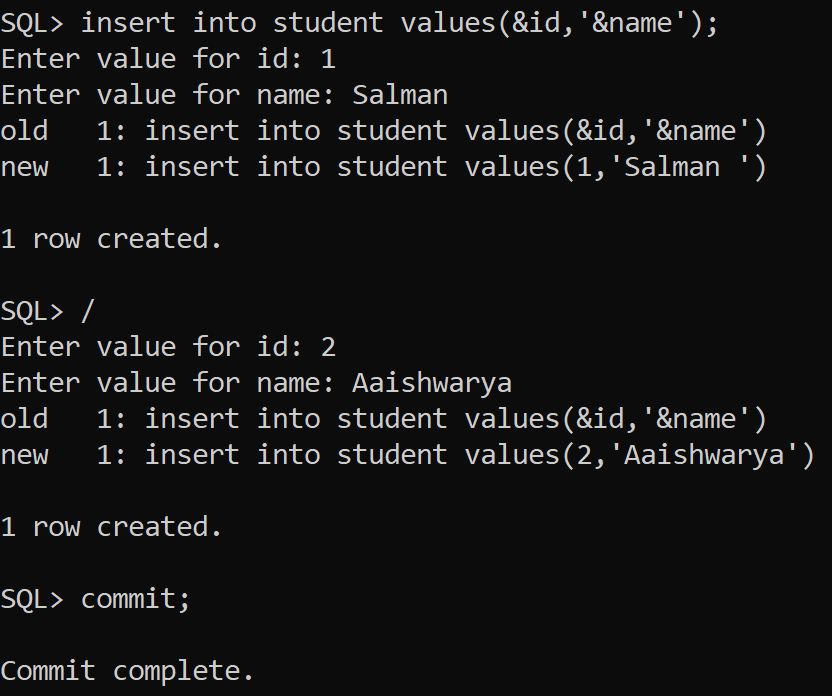
alter table student add constraint student\_pk primary key(id);

**Output:**



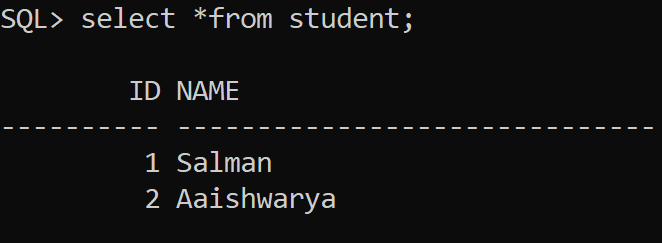
**Syntax:**  insert into student values (&id,’&name’);

**Output:**



**Syntax:**  select \*from student;

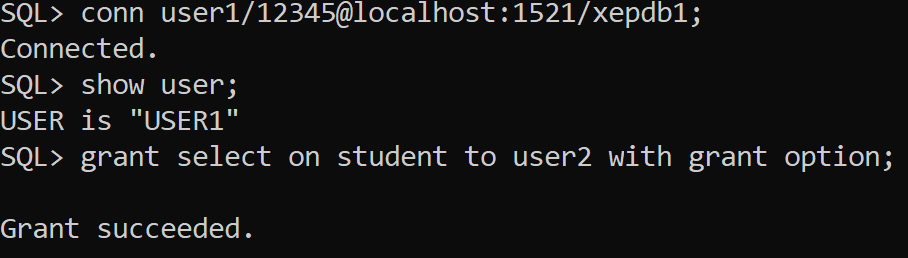
**Output:**



* + 1. Granting SELECT privilege and test the effect. Also do practice of passing object privileges with grant option.

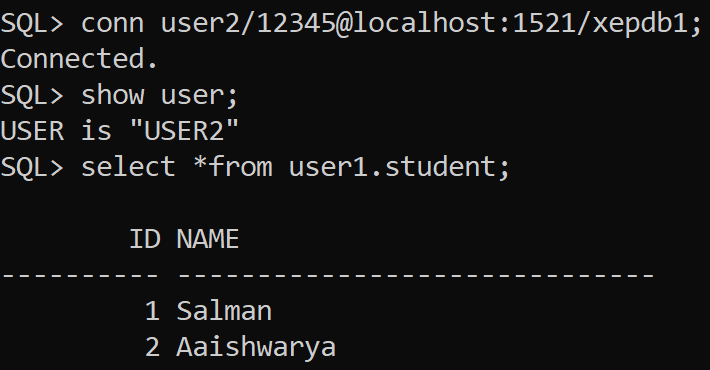
**Syntax:**  grant select on student to user2 with grant option;

**Output:**



**Syntax:**  select \*from user1.student;

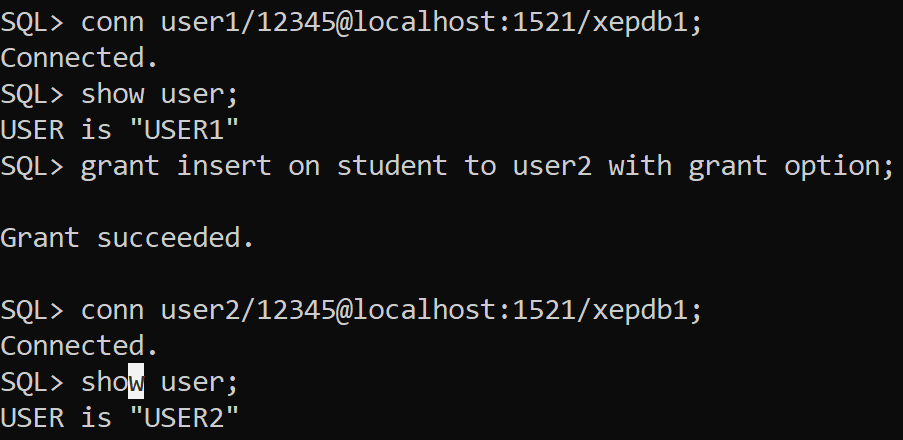
**Output:**



* + 1. Granting INSERT privileges and inserts some rows from another user.

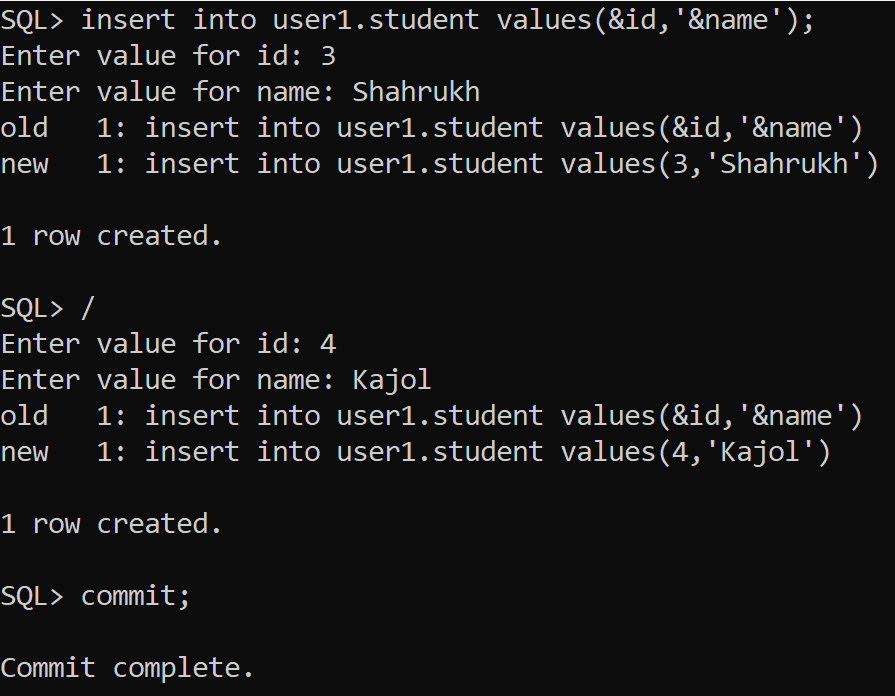
**Syntax:**  grant insert on student to user2 with grant option;

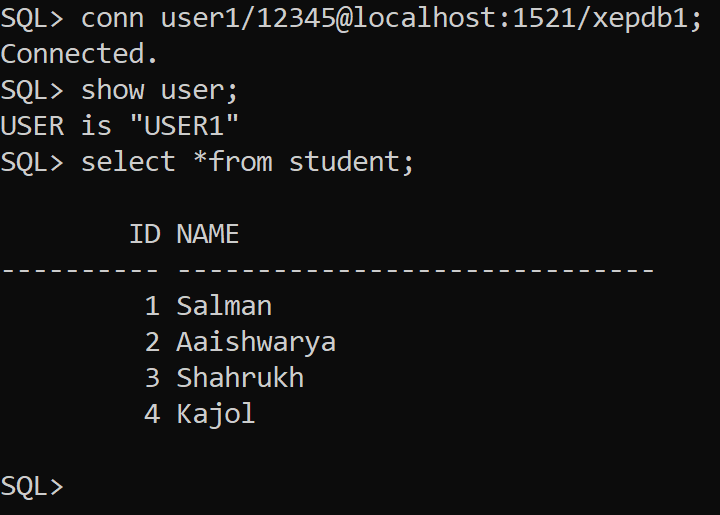
**Output:**



**Syntax:**  insert into user1.student values(&id,’&name’);

**Output:**

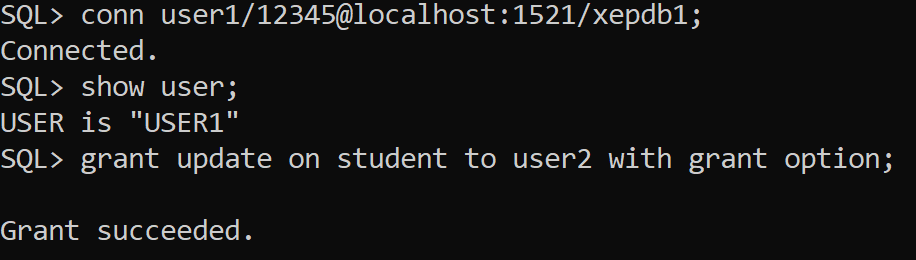




* + 1. Grant update privilege on table for specified column only

**Syntax:**  grant update on student to user2 with grant option;

**Output:**

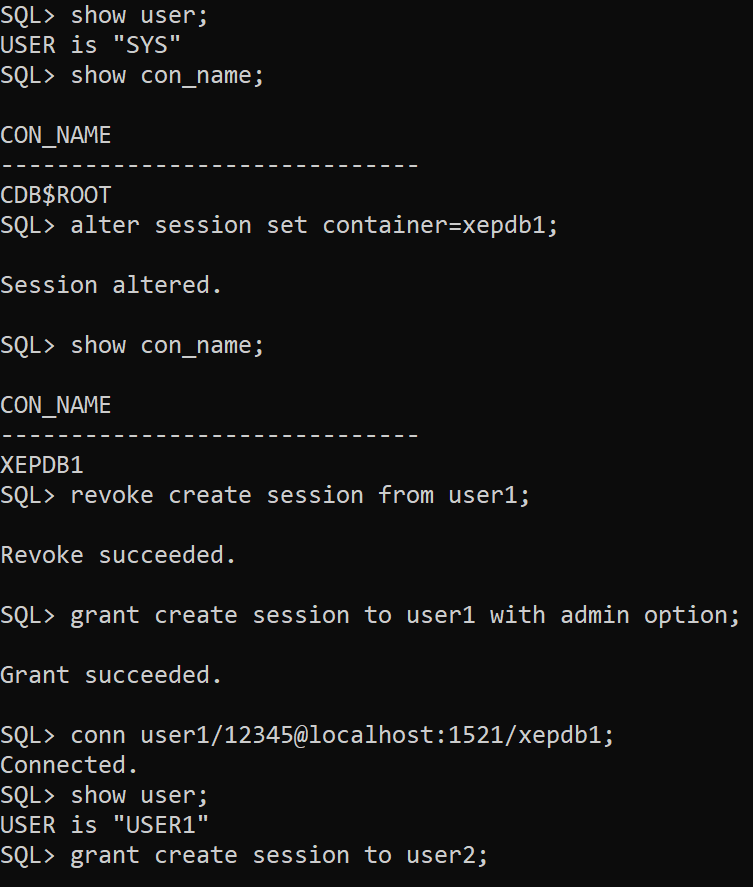


Update table name set col name =new value where id =1

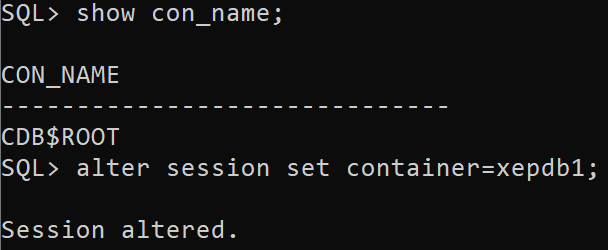
* + 1. Grant system privileges to the user and pass the system privileges to other user with admin option and observe the impact.

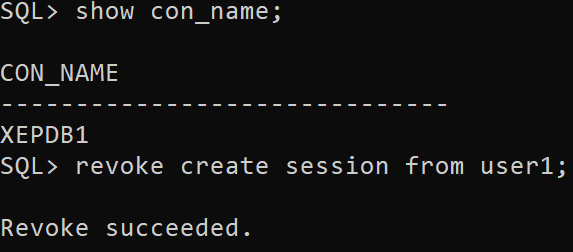
**Syntax:**  grant create session to user1 with admin option;

**Output:**



* 1. **Role**
     1. Create a Role and grant privileges to roles and grant roles to the users.

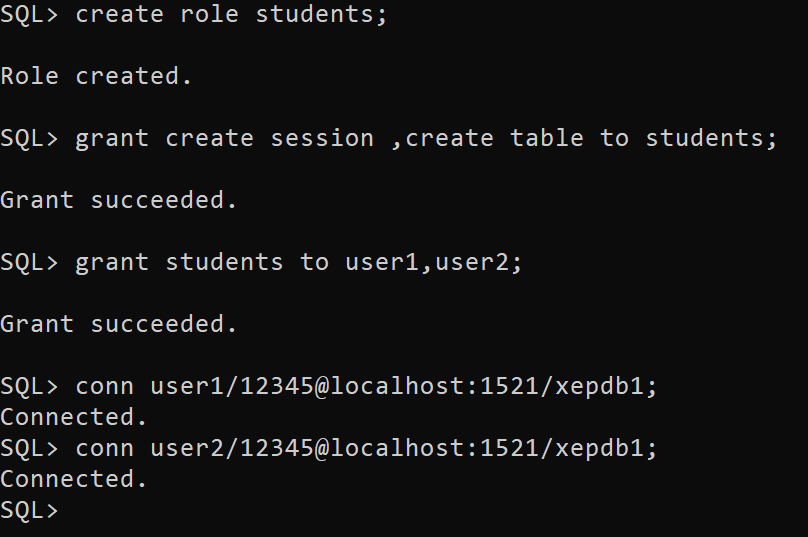




**Syntax:**  create role students;

grant students to user1,user2;

**Output:**



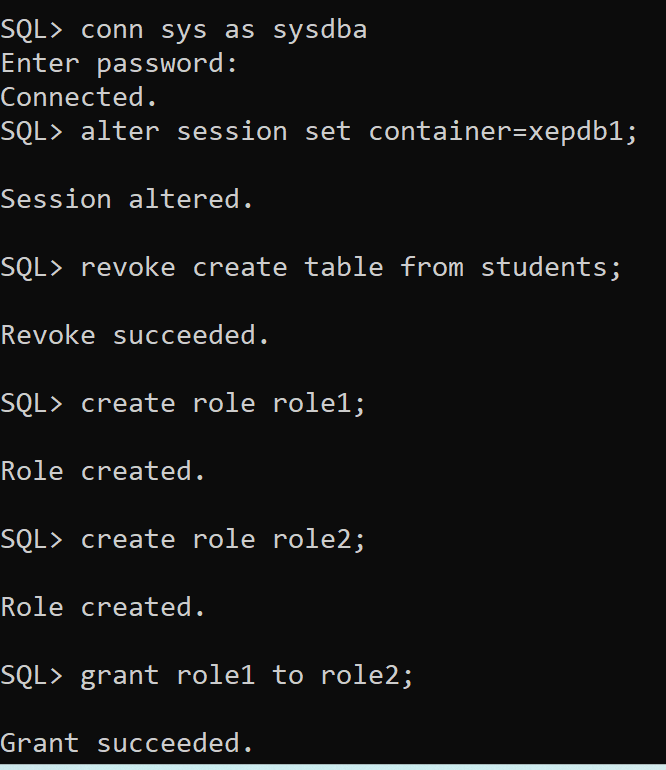
* + 1. Grant role to other role and assign role to user and test the impact

**Syntax:**  create role role1;

create role role2;

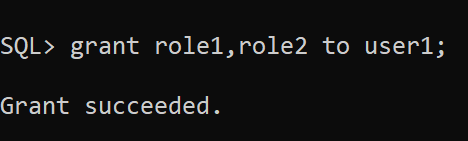
grant role1 to role2;

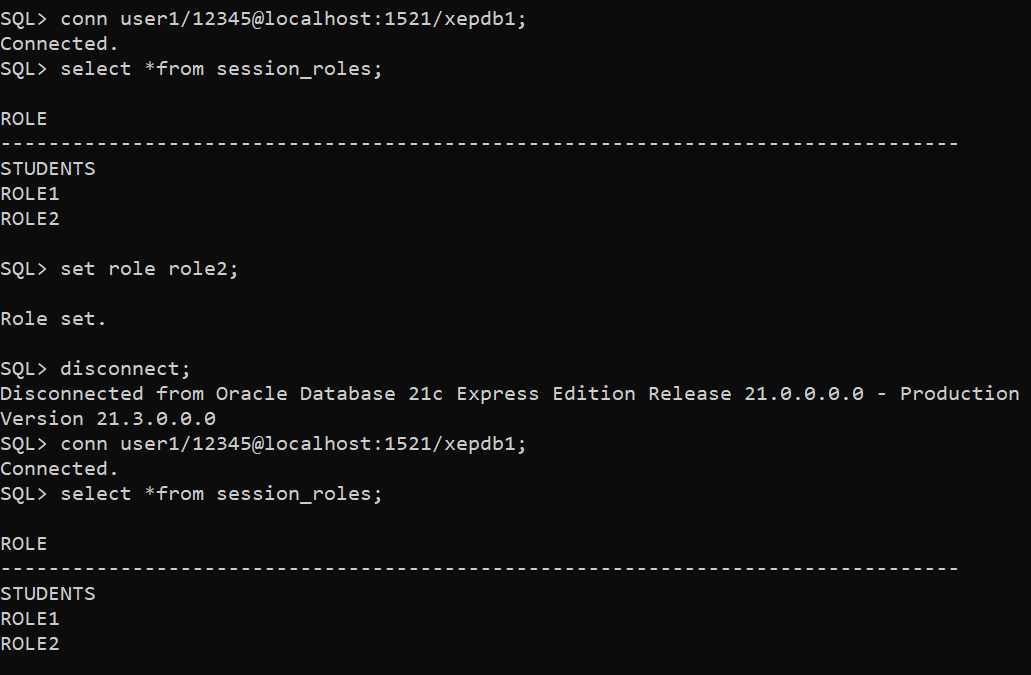
**Output:**



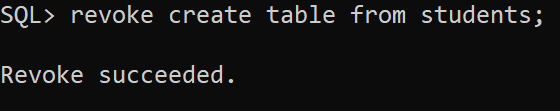
**Syntax:** grant role1, role2 to user1;

**Output:**





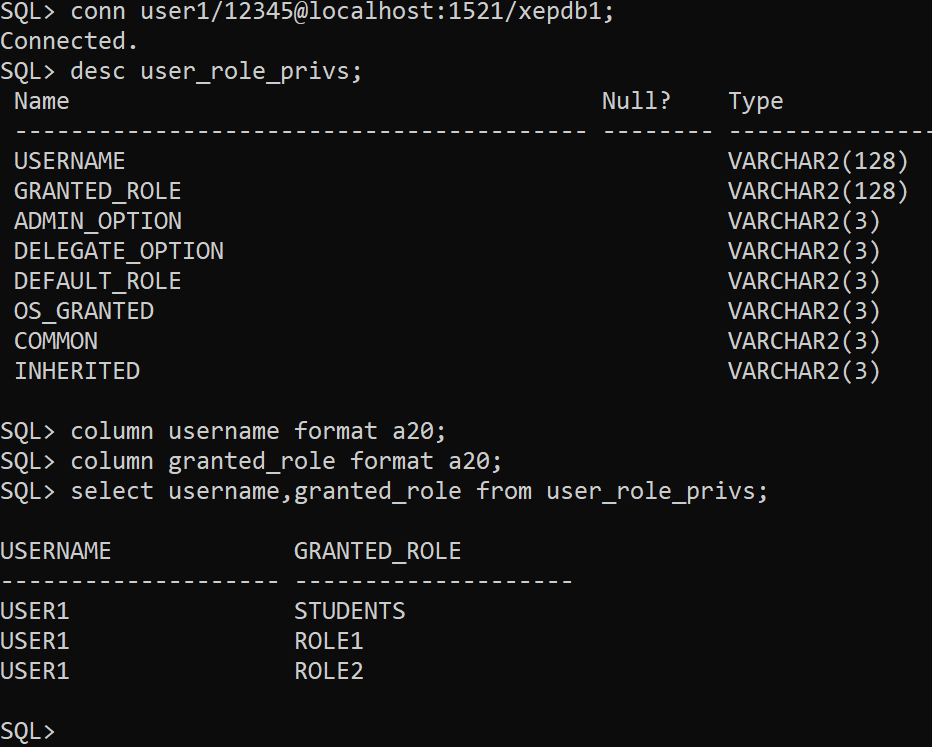
* + 1. Revoke system/object privileges granted to roles and users



* 1. **Data dictionary views to list the privileges**
     1. List the system privileges, object privileges and roles granted/received by the user

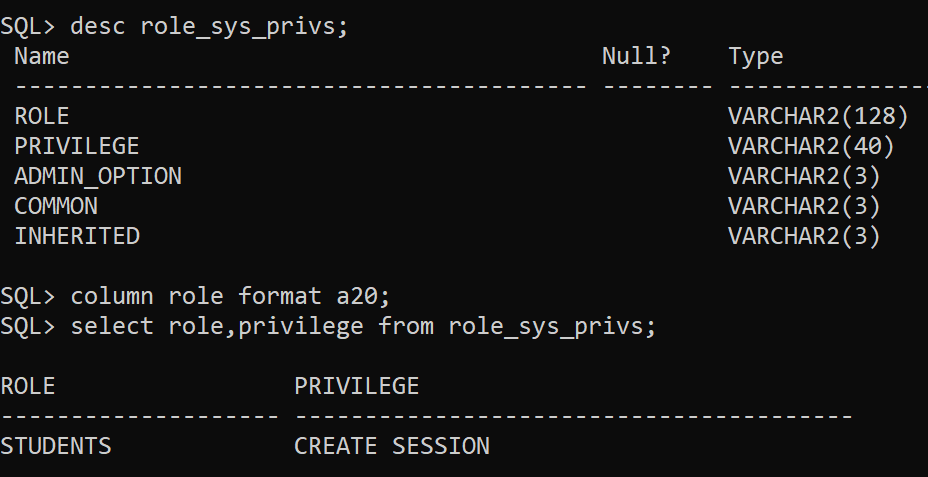
**Syntax:** desc user\_role\_privs;

**Output:**



**Syntax:** desc role\_sys\_privs;

**Output:**



* 1. **Profile**
     1. Create Profile with following parameter and assign the profile to the user

SESSIONS\_PER\_USER 2

FAILED\_LOGIN\_ATTEMPTS 3

IDLE\_TIME 2 (in minutes)

CONNECT\_TIME 5 (in minutes)

PASSWORD\_LIFE\_TIME 60

PASSWORD\_GRACE\_TIME 10

PASSWORD\_REUSE\_TIME 1

PASSWORD\_LOCK\_TIME 7

PASSWORD\_REUSE\_MAX 3;

**Syntax:**

SQL> create profile lab\_profile

2 limit

3 sessions\_per\_user 2

4 failed\_login\_attempts 3

5 idle\_time 2

6 connect\_time 5

7 password\_life\_time 10

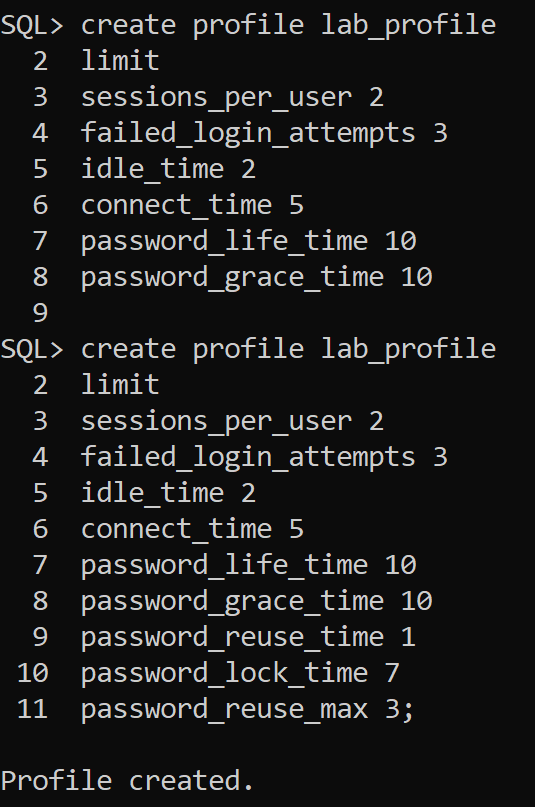
8 password\_grace\_time 10

9 password\_reuse\_time 1

10 password\_lock\_time 7

11 password\_reuse\_max 3;

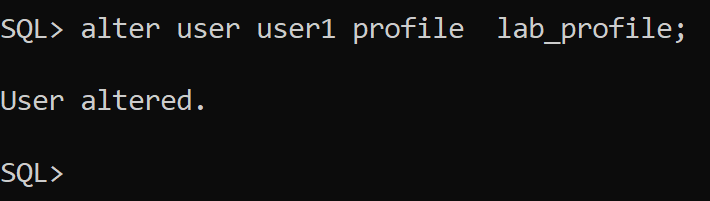
**Output:**



* + 1. Apply the profile by setting the resource\_limit parameter true

**Syntax:** alter user user1 profile lab\_profile;

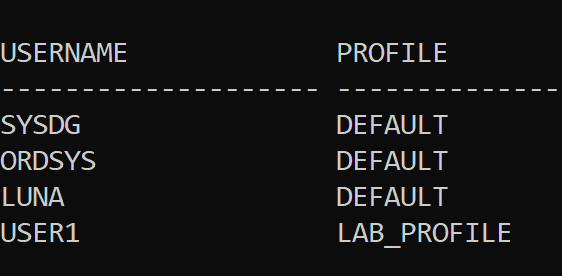
**Output:**



* + 1. Alter the profile and check the impact of it

**Syntax:** select username,profile from dba\_users;

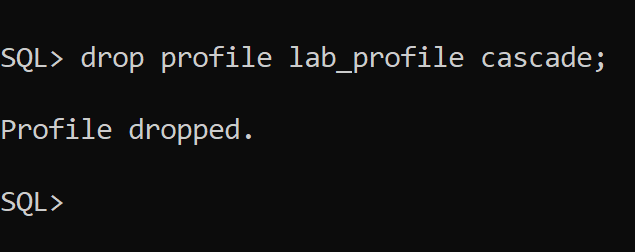
**Output:**



* + 1. Delete the profile and check the user profile assigned after profile deleted

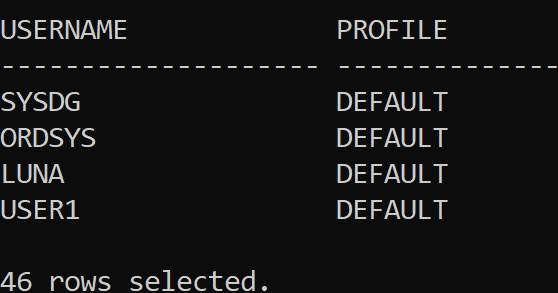
**Syntax:** drop profile lab\_profile cascade;

**Output:**



**Syntax:** select username,profile from dba\_users;

**Output:**

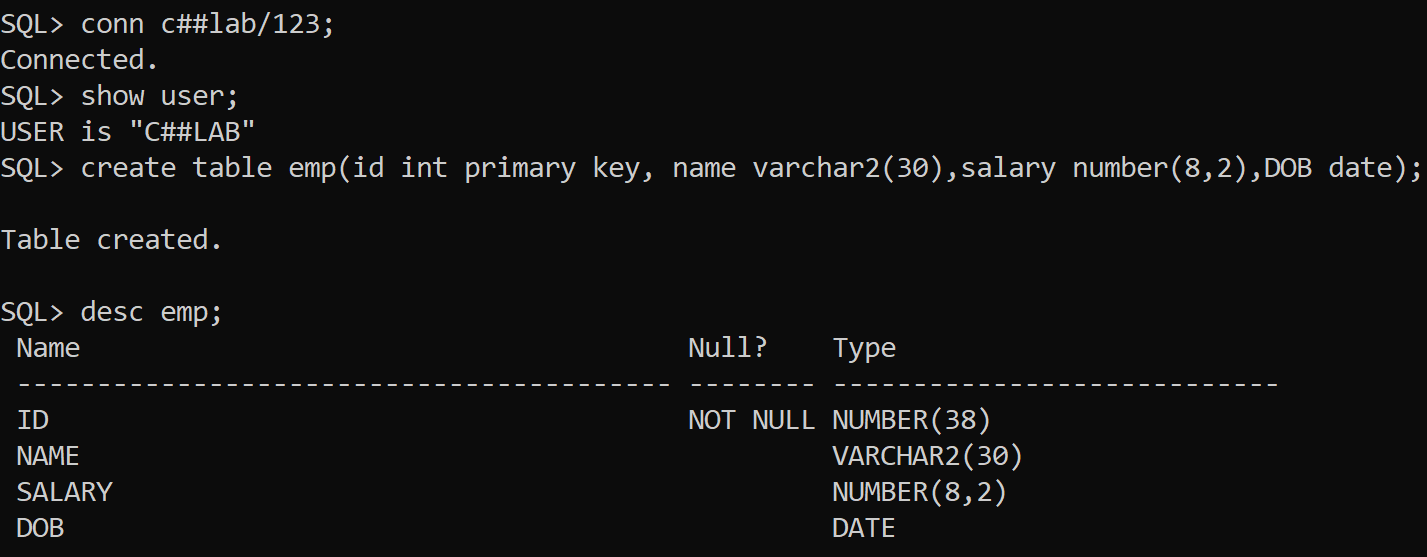


**Lab Assignment#2**

1. Creating oracle objects:
   1. Table:
      1. Create table with different datatype with primary key

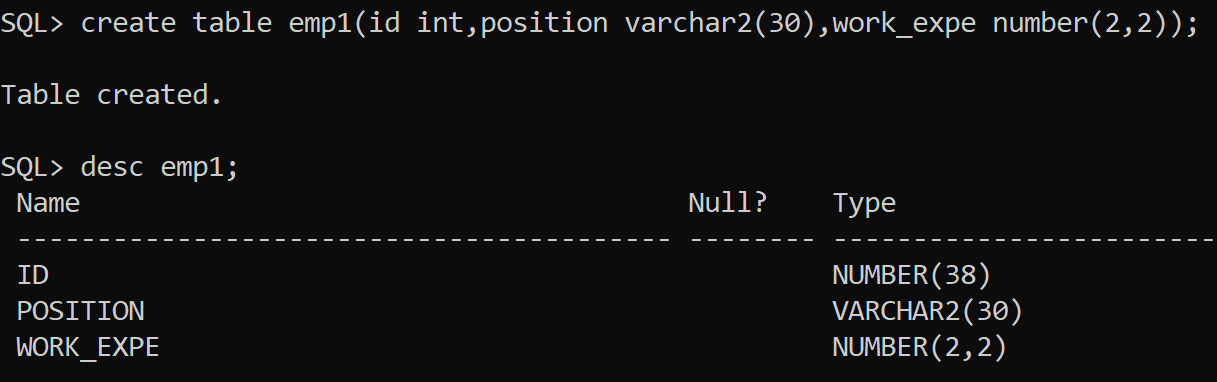
**Syntax**: create table emp(id int primary key, name varchar2(30),salary number(8,2),DOB date);

**Output:**



**Syntax**: create table emp1(id int , position varchar2(30),work\_expe(2,2));

**Output:**



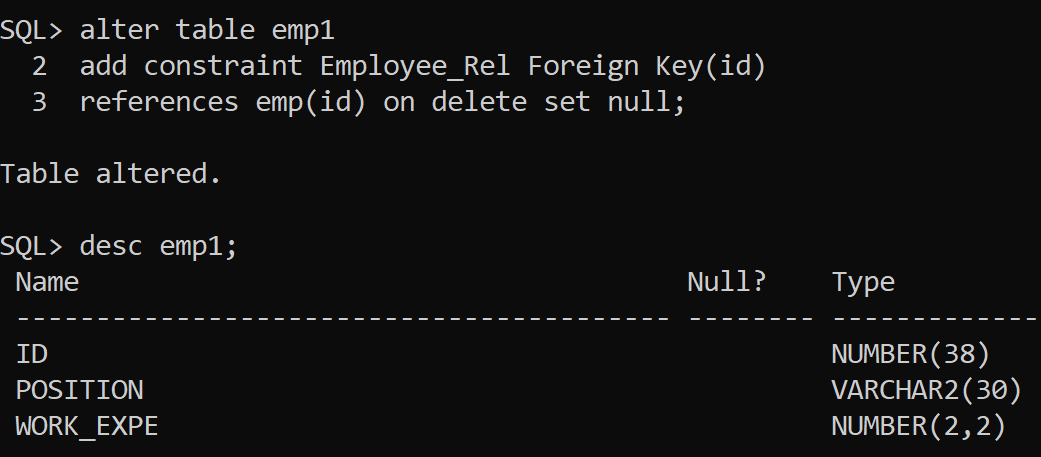
* + 1. Add foreign key and check constraint

**Syntax**: alter table emp1

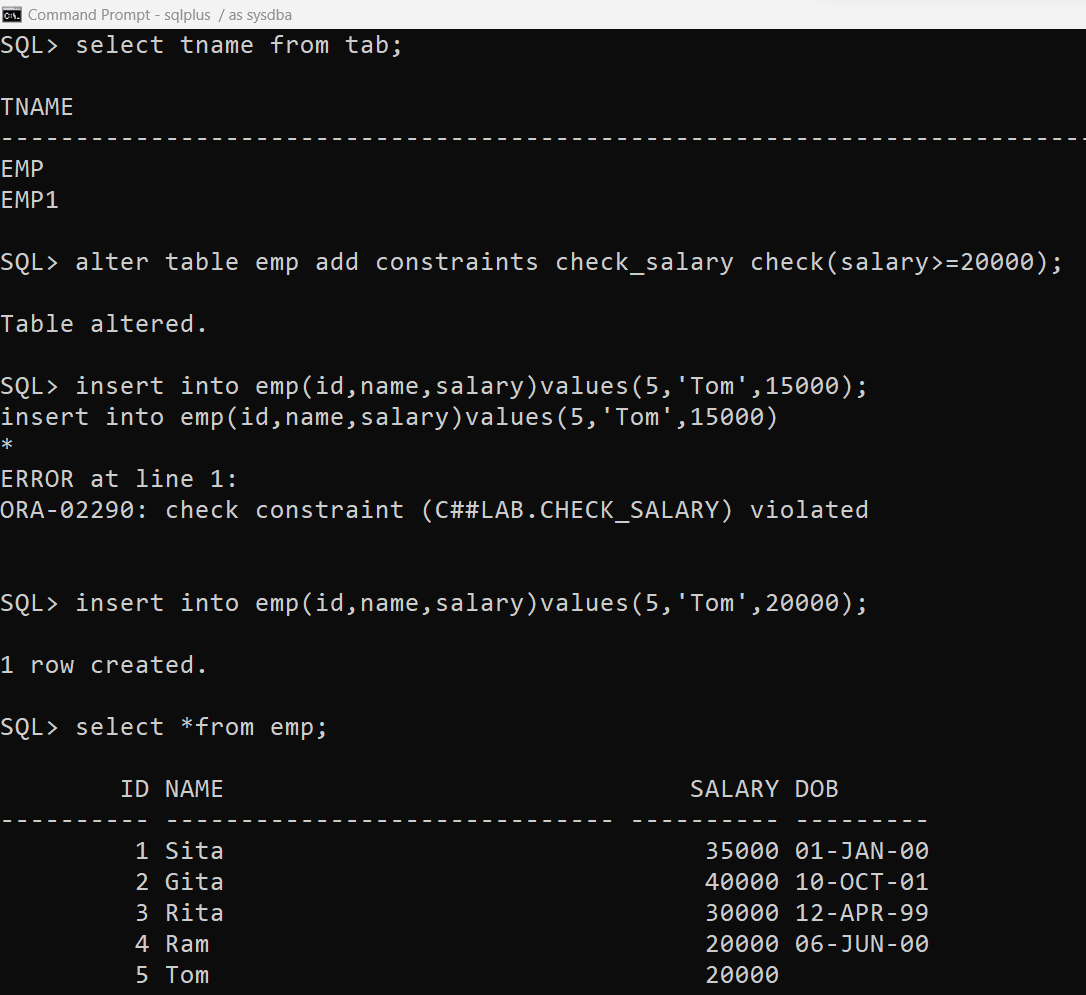
2 add constraint Employee\_Rel Foreign Key(id)

3 references emp(id) on delete set null;

**Output:**

  
**Syntax**: alter table emp add constraints check\_salary check(salary>=20000);

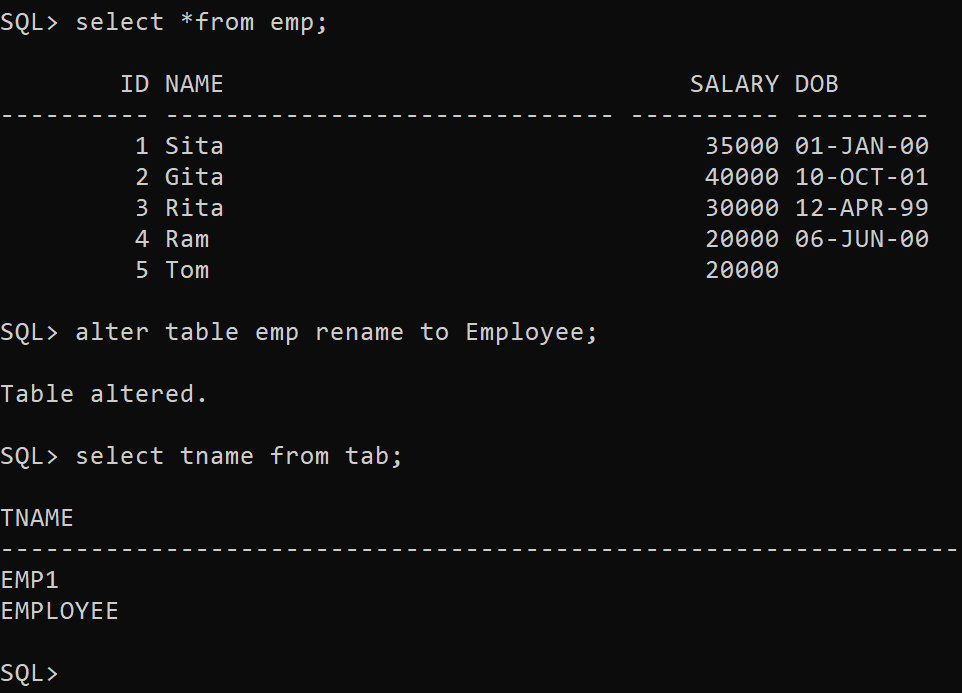
**Output:**



* + 1. Rename table and renaming table column

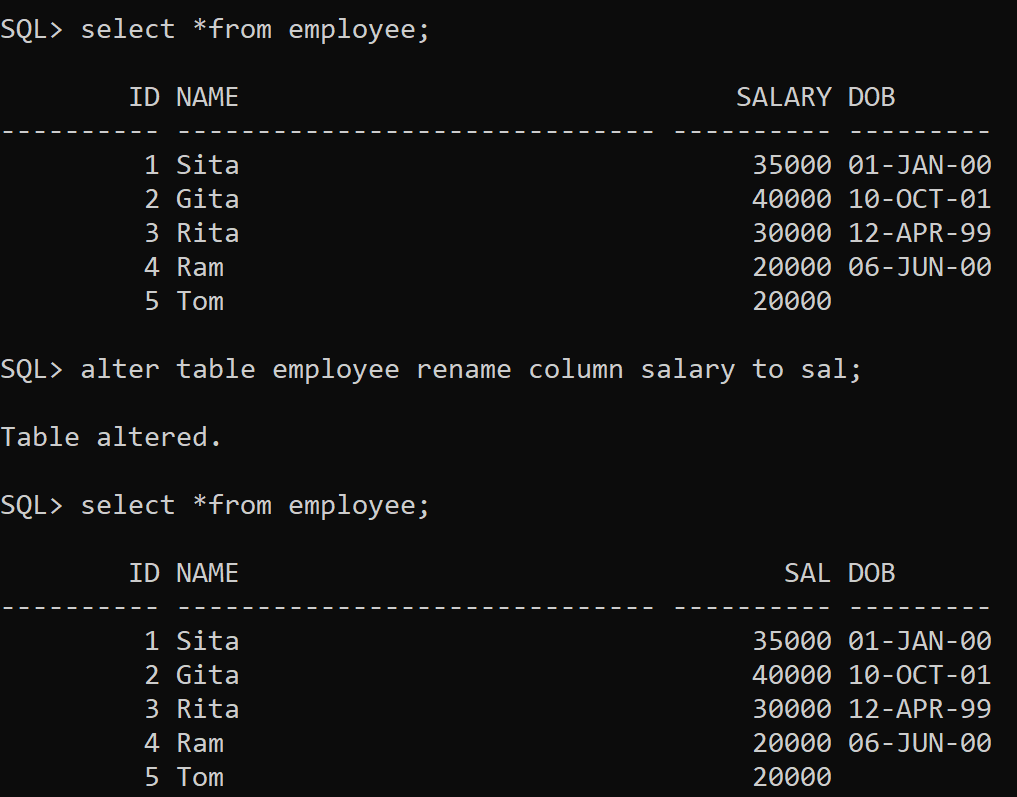
**Syntax**: alter table emp rename to Employee;

**Output:**



**Syntax**: alter table employee rename column salary to sal;

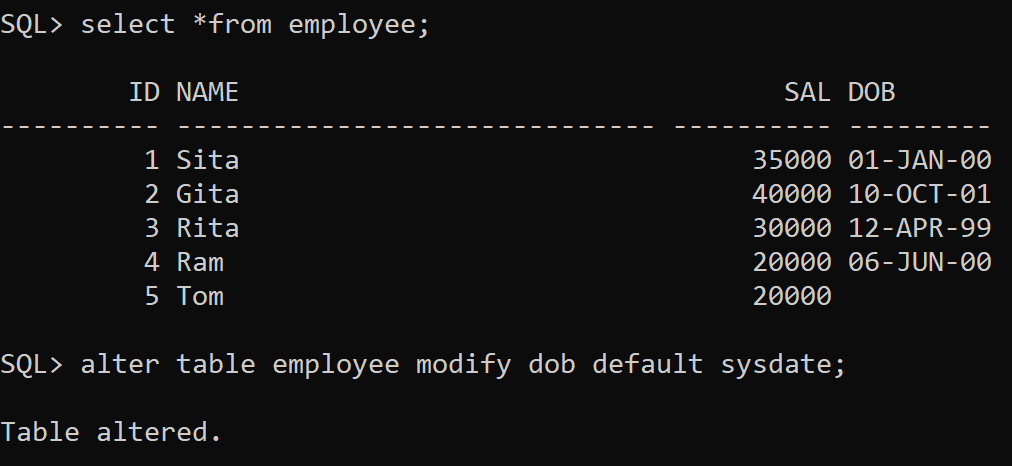
**Output:**

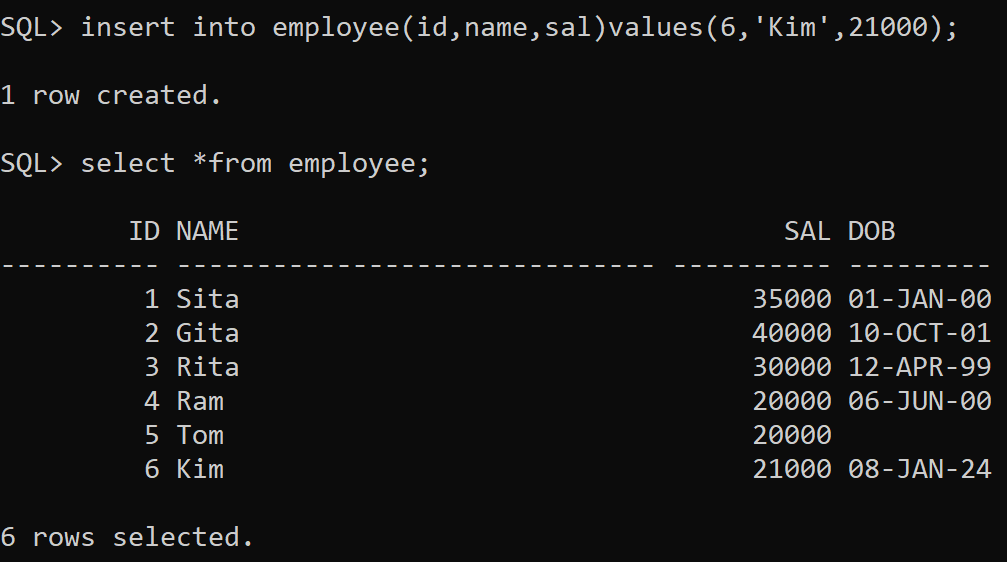


* + 1. Modify the datatype of a column

**Syntax**: alter table employee modify dob default sysdate;

**Output:**

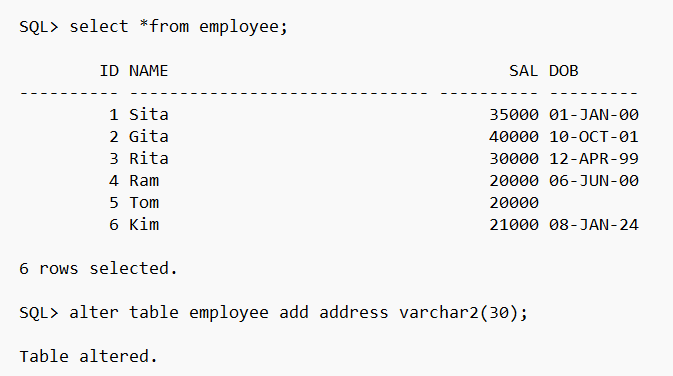


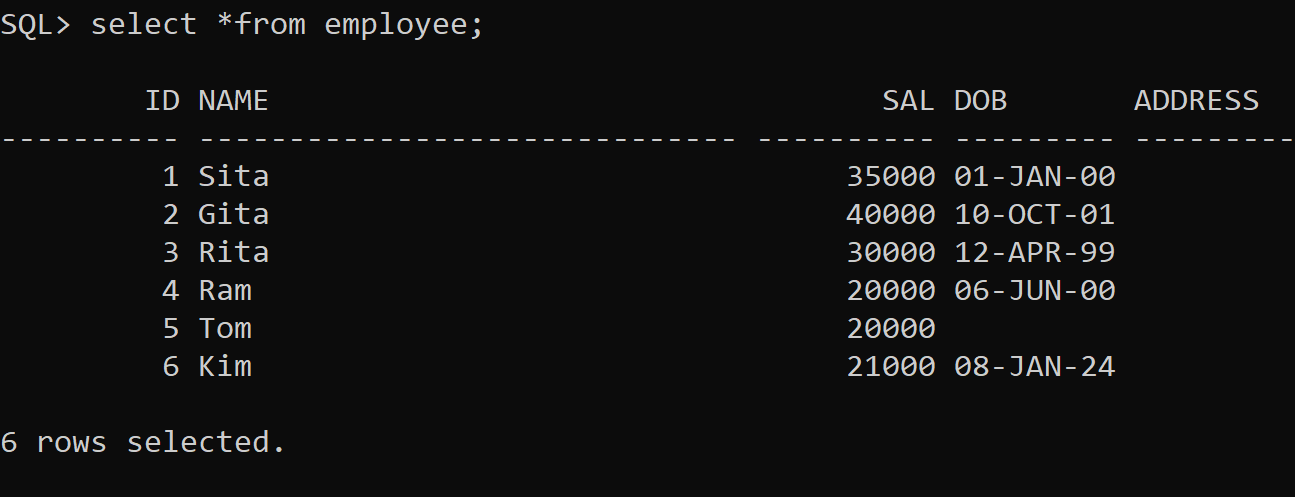


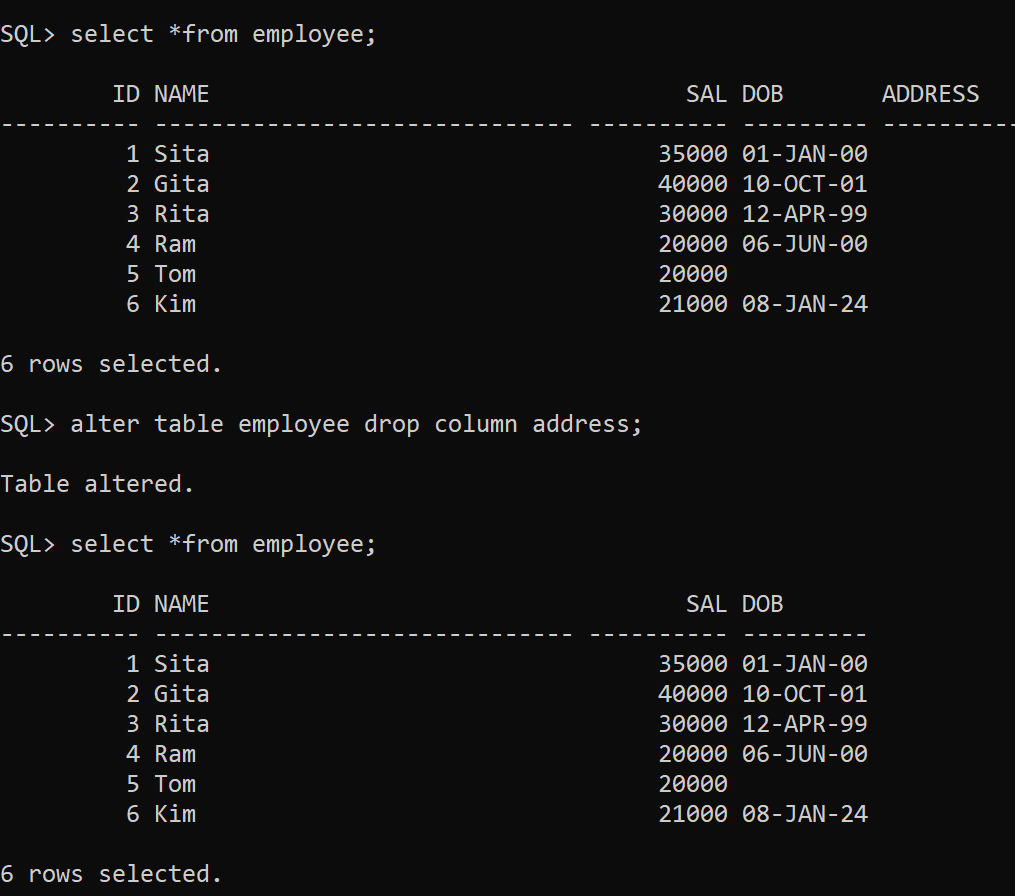
* + 1. Add and drop new field in a table

**Syntax**: alter table employee add address varchar2(30);

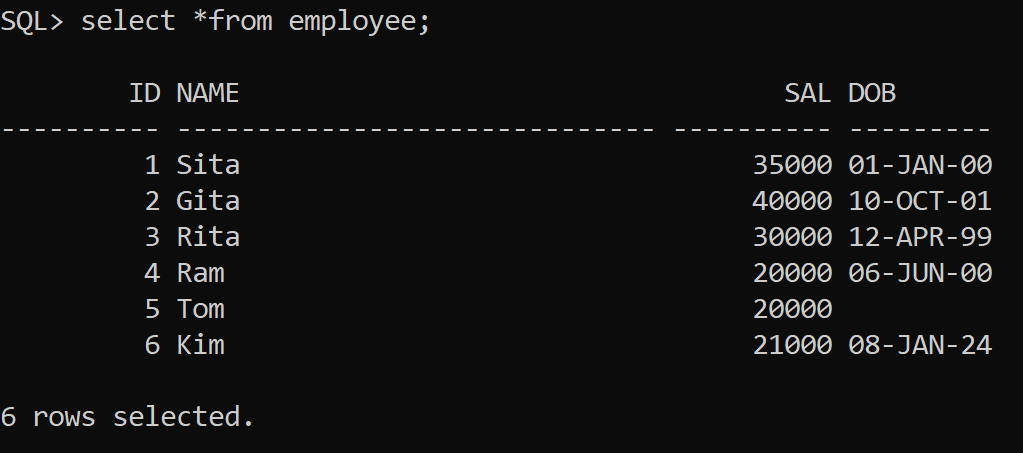
**Output:**

****





* + 1. Insert data in a table using dynamic input



set serveroutput on;

**declare**

v\_id int := 7;

v\_name varchar(30) := 'Luna';

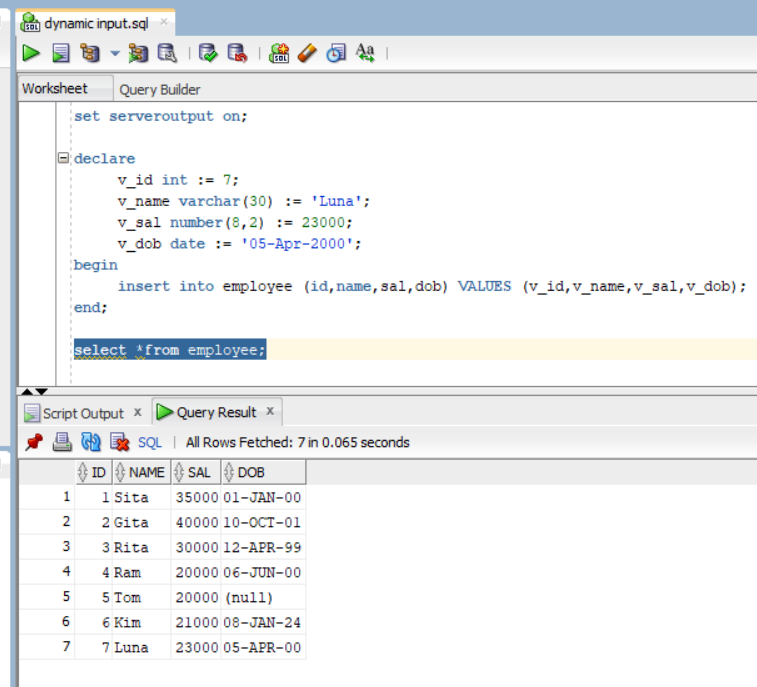
v\_sal number(8,2) := 23000;

v\_dob date := '05-Apr-2000;

**begin**

insert into employee (id,name,sal,dob) VALUES (v\_id,v\_name,v\_sal,v\_dob);

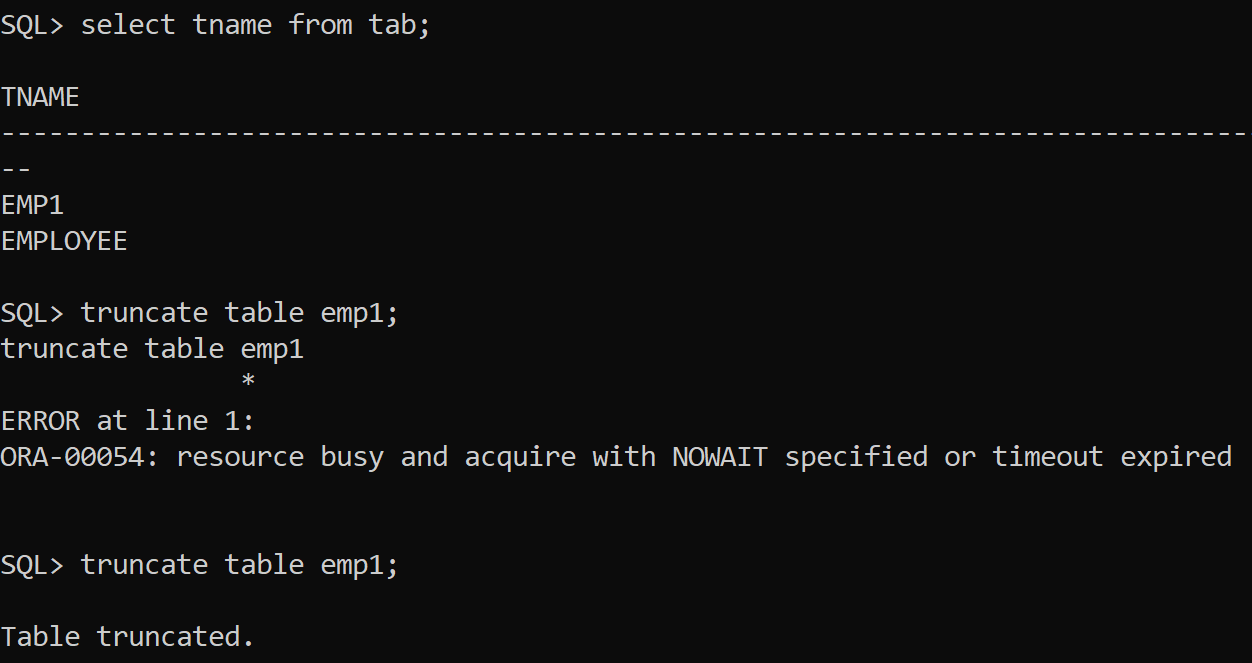
**end;**



* + 1. Truncate and drop the table

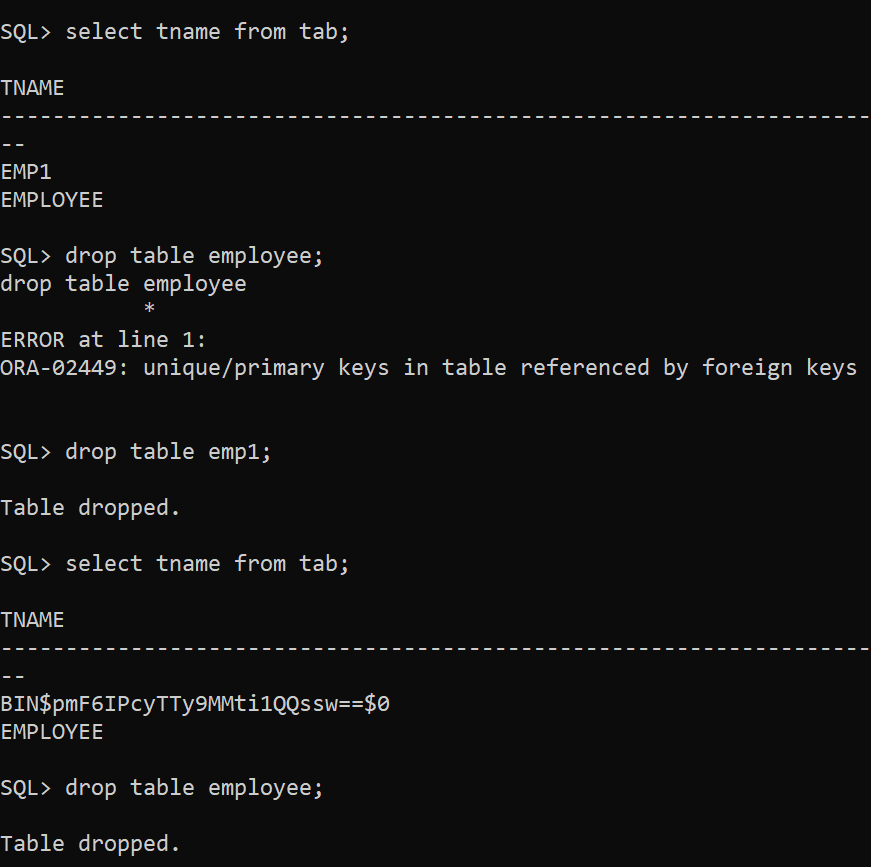
**Syntax**: truncate table emp1;

**Output:**



**Syntax**: drop table table\_name;

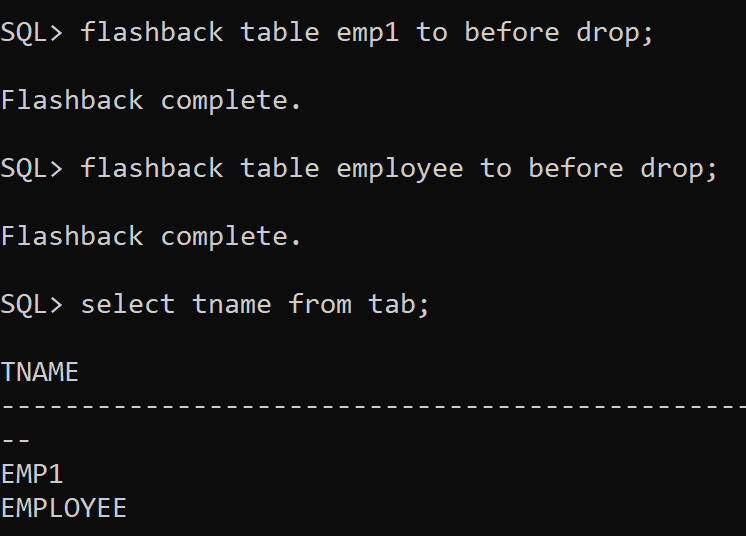
**Output:**



* + 1. Flashback the table from recycle bin

**Syntax**: flashback table table\_name to before drop;

**Output:**

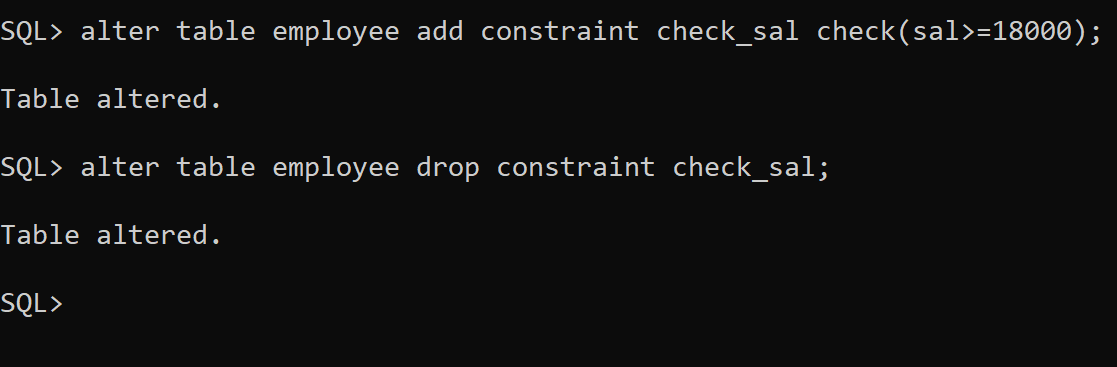


* + 1. Add and drop the constraint in a table

**Syntax**: alter table employee add constraint check\_sal check(sal>=18000);

alter table employee drop constraint check\_sal;

**Output:**



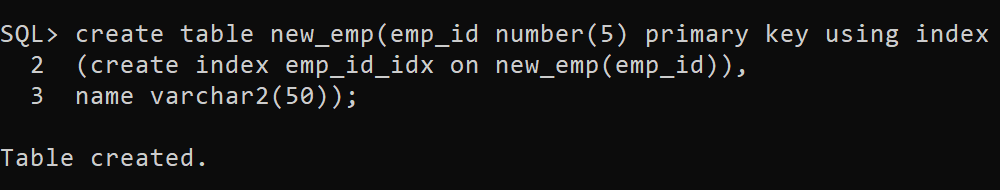
* 1. **Index**
     1. Create index while creating table

**Syntax**: create table new\_emp(emp\_id number(5) primary key using index

(create index emp\_id\_idx on new\_emp(emp\_id)),

name varchar2(50));

**Output:**

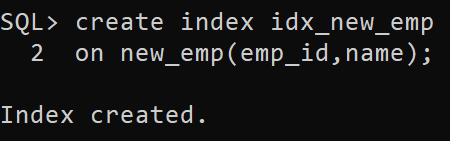


* + 1. Create composite index in a table

**Syntax**: create index idx\_new\_emp

on new\_emp(emp\_id,name);

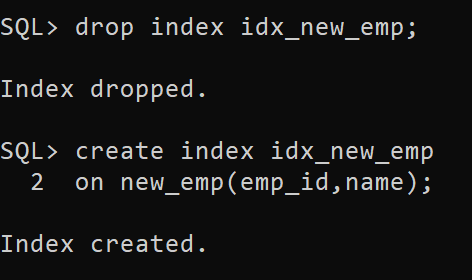
**Output:**



* + 1. Modify the index (drop and recreate)

**Syntax**: drop index idx\_new\_emp;

**Output:**



* 1. **Synonym** (what is synonym and write the advantage of it)

*What is synonym?*

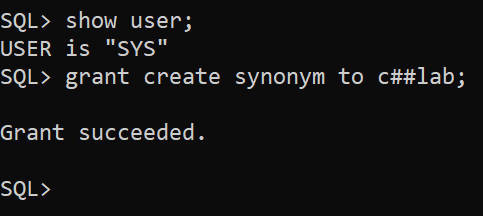
* A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects.

*Advantages of synonym*

* Synonym is an alias for a schema object
* Synonym can create a level of security by hiding the owner's name and exact object name
* Synonyms are convenient to use and reduces the complexity of the SQL statements for the database users.
* You generally use synonyms when you are granting access to an object from another schema, and you don't want the users to have to worry about knowing which schema owns the object.
* When you grant object privilege on a synonym, you are really granting privileges on the underlying object, and the synonym is acting only as an alis of the object in the GRANT statement.
  + 1. Grant select on table to different user and create private synonym

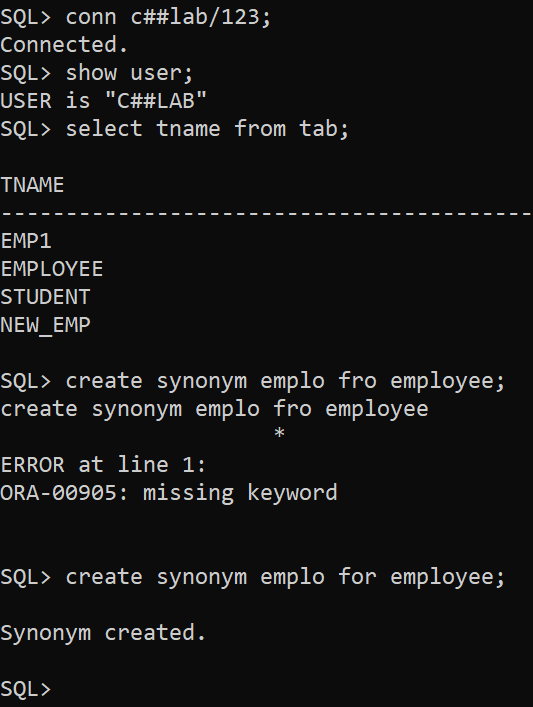
**Syntax**: grant create synonym to c##lab;

**Output:**



**Syntax**: create synonym emplo for employee;

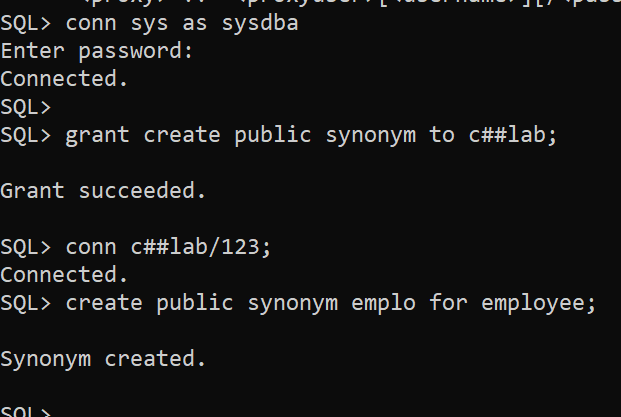
**Output:**



* + 1. Create public synonym and grant it to the other user

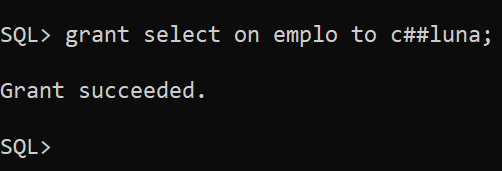
**Syntax**: create public synonym emplo for employee;

**Output:**

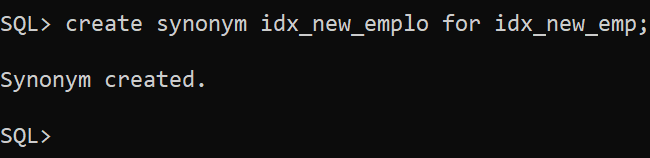


**Syntax**: grant select on employ to c##luna;

**Output:**



* + 1. Create synonym for other object like index, procedure etc



* 1. **Sequence** (what is sequence? why it is used?)

*What is sequence?*

* A sequence is an object in Oracle that is used to generate a number sequence.

*Why it is used?*

* In Oracle, we can create an autonumber field by using sequences.
* This can be useful when we need to create a unique number to act as a primary key.
  + 1. Create sequence and apply it to the insert statement in table

**Syntax**: create sequence emp\_seq

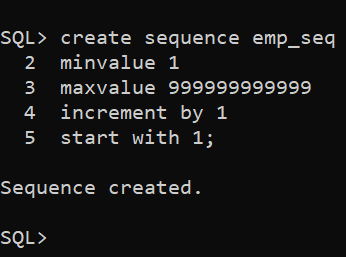
minvalue 1

maxvalue 999999999999

increment by 1

start with 1;

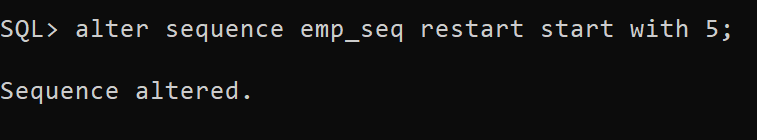
**Output:**



* + 1. Alter sequence

**Syntax**: alter sequence emp\_seq restart start with 5;

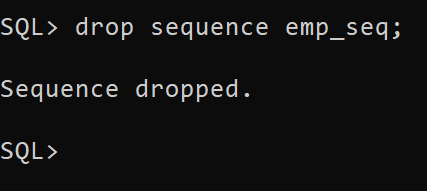
**Output:**



* + 1. Drop sequence

**Syntax**:drop sequence emp\_seq;

**Output:**



* 1. **View** (what is view? Write advantage of using views)

*What is view?*

* In Oracle, view is a virtual table that does not physically exist.
* It is stored in Oracle data dictionary and do not store any data. It can be executed when called.
* A view is created by a query joining one or more tables.
* We can query views and even modify, remove or add data using update, delete or insert statements, provided the user has the appropriate privileges on the underlying base tables.
* We need to have a create view privilege to create a view.

*Write advantage of using views.*

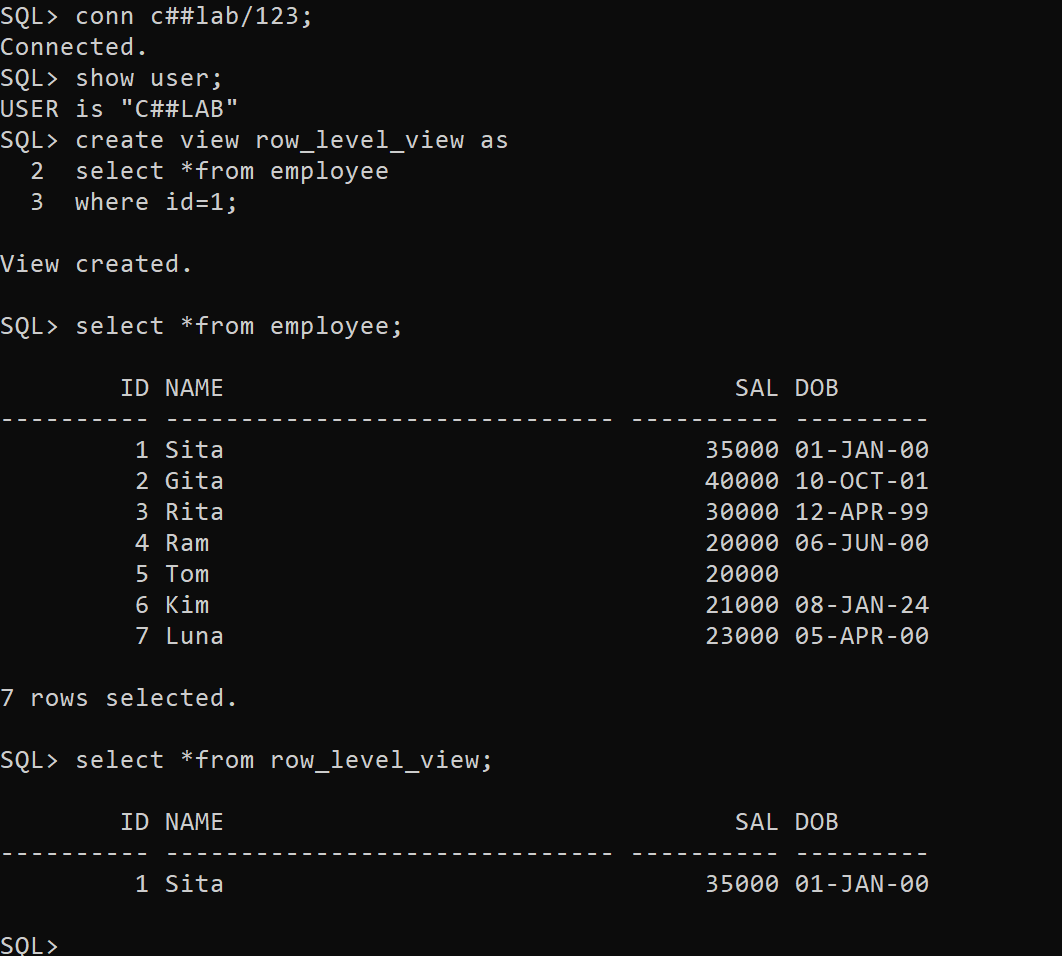
* views are used in application for several reasons; including the following:
* reduce complexity
* improve security
* increase convenience
* rename table columns
* customize the data for user
* protect data integrity
* We can use a view to add column level or value-based security to table
* Column level security is provided by creating a view that gives access to selected columns of base tables
* value based security involves using where clause in the view definition which displays only selected rows of base tables
  + 1. Create row level and column level views

**Syntax**: create view row\_level\_view as

select \*from employee

where id=1;

**Output:**

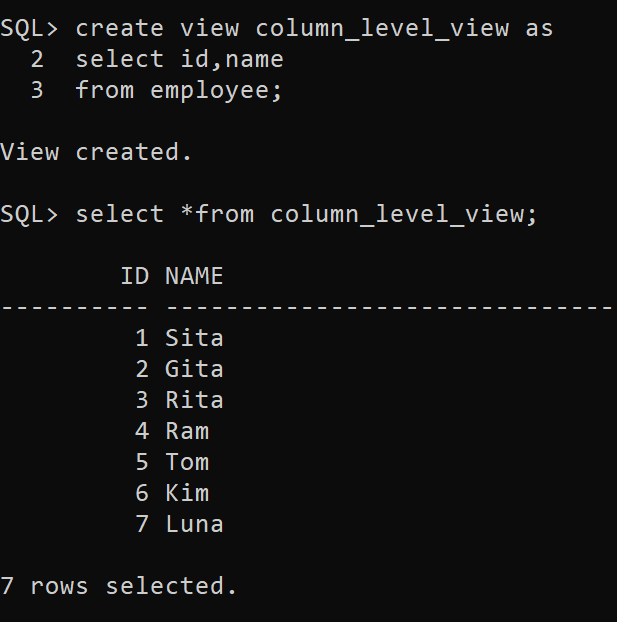


**Syntax**: create view column\_level\_view as

select id,name

from employee;

**Output:**



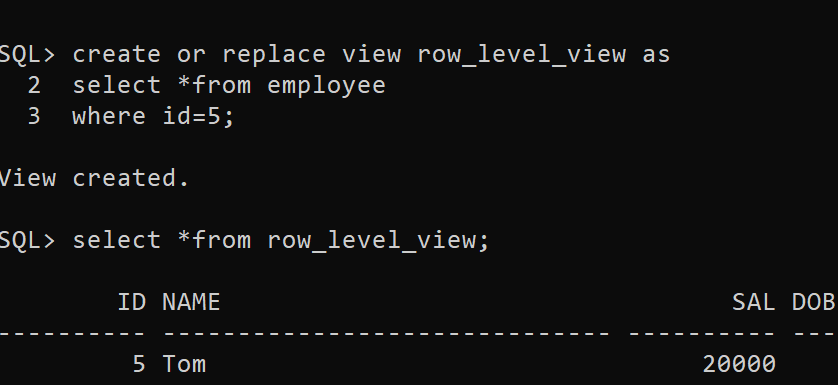
* + 1. Modify the view definition

**Syntax**: create or replace view row\_level\_view as

select \*from employee

where id=5;

**Output:**



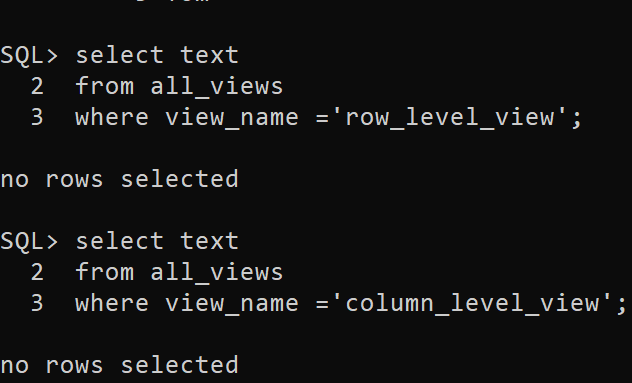
* + 1. Display the underlying sql of view definition

**Syntax:** create or replace view row\_level\_view as

select \*from employee

where id=5;

**Output:**

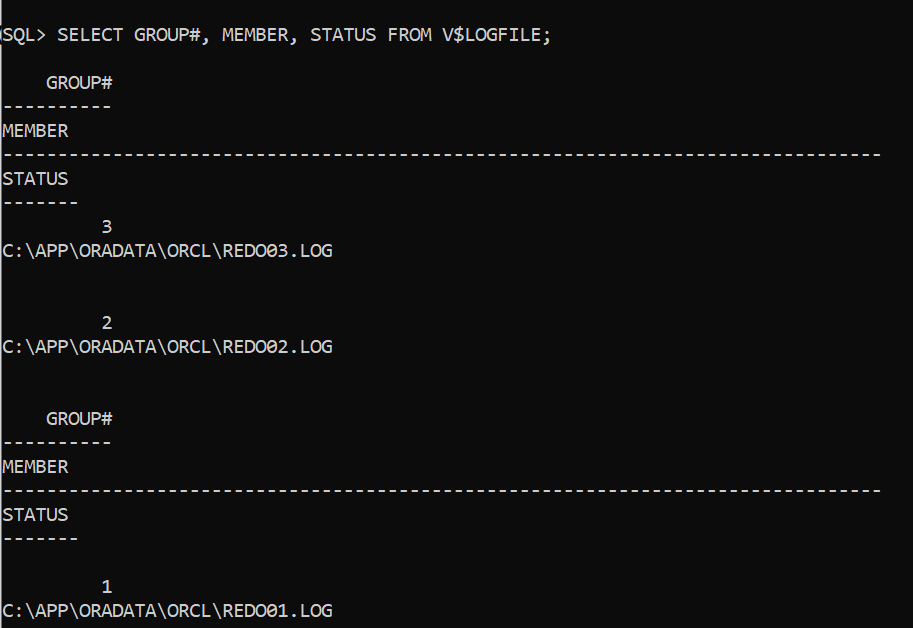


**Lab Assignment#3**

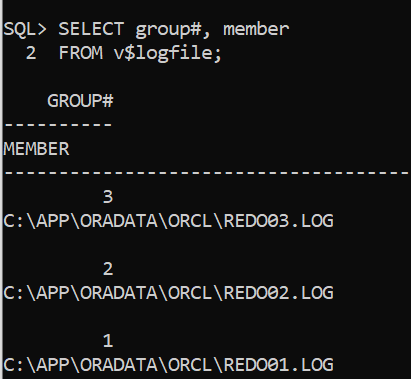
1.What is multiplexing and write command to Multiplex the Redo Log Files for your database.

Multiplexing in the context of database management refers to the practice of creating multiple identical copies of critical components such as redo log files or control files. This redundancy helps in ensuring data integrity and availability in case of failures.

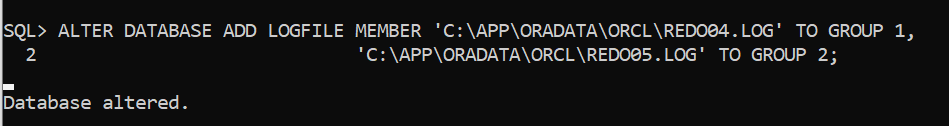
* 1. Show the groups, members, status and other related information about redo logs



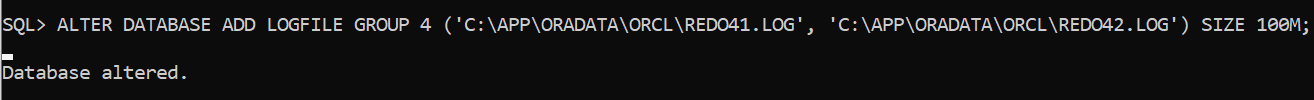
* 1. Determine where to locate the files for each group



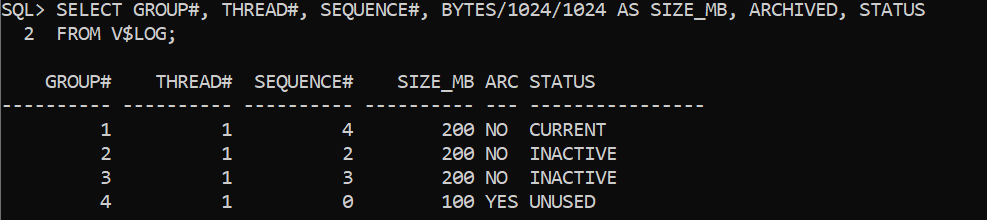
* 1. Add an additional Redo Log File to each of the two existing Redo Log Groups for your database



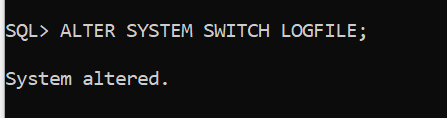
* 1. Add an additional Redo Log Group with two Redo Log Files in the group



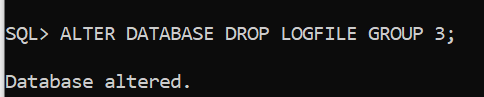
* 1. Query both V$LOG and V$LOGFILE to verify the result



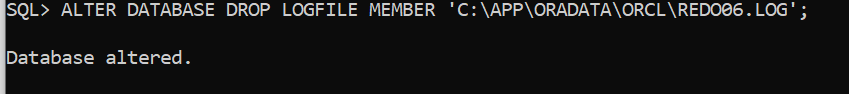
* 1. Switch the log file manually



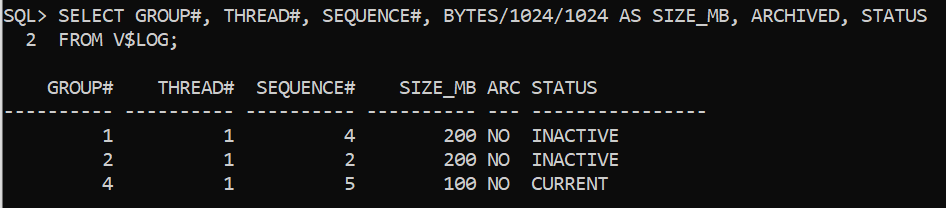
* 1. Delete one of the redo log group



* 1. Delete one of the member from existing group

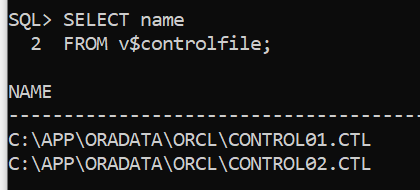


* 1. Observe the different status of the redo logs



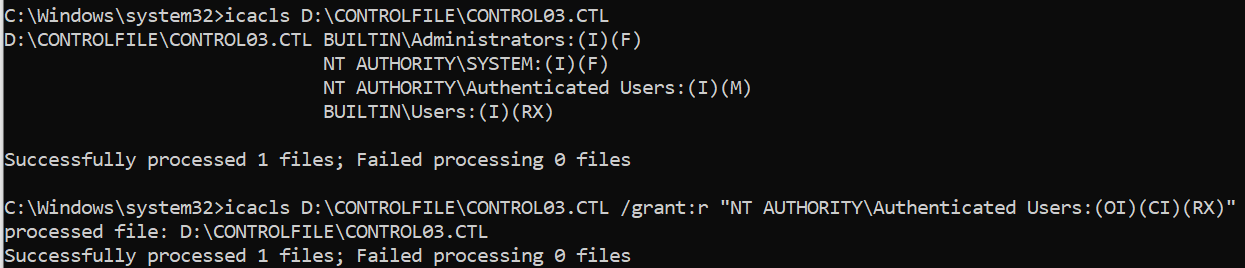
2. Add a new control file (Multiplex the Control File)

1. Locate the control files in your database

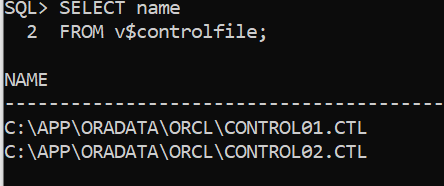


1. Add a second control file (for example, control02.ctl).  Before you do this, check to see if you’ve already created 2 control files.  If you have, then you will be adding a third control file instead of adding a second one. Multiplex the control files by spreading them across the disk drives that are available to you. Check the permission settings on the new control file—since you will be the owner, the DBA group that includes the Oracle user will need permission to access the file – a setting of 660 should be sufficient (you are allowed to try other permission settings as long as the new control file is accessible when the database is restarted).

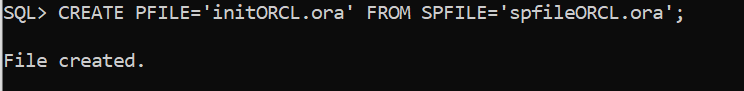




1. Startup the database.  Query the V$CONTROLFILE view to display the names/locations of your control files



3.Create static parameter file (pfile) from dynamic parameter file (spfile)



**Lab Assignment#4**

1.Tablespace: What is tablespace? how it is related with datafile?

A tablespace in Oracle Database is a logical storage container used to organize and manage the storage of database objects such as tables, indexes, and other schema objects. It serves as a logical unit of database storage and management.

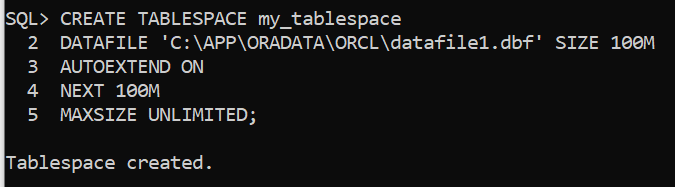
Tablespaces are essential for database administration, as they provide a way to allocate and manage disk space for database objects. Each tablespace consists of one or more physical datafiles, which are actual files on the filesystem where the database objects are stored.

Tablespace: A tablespace is a logical storage container that groups related database objects together.

Datafile: A datafile is a physical file on the filesystem that belongs to a tablespace. Each tablespace consists of one or more datafiles where the actual data for the database objects is stored.

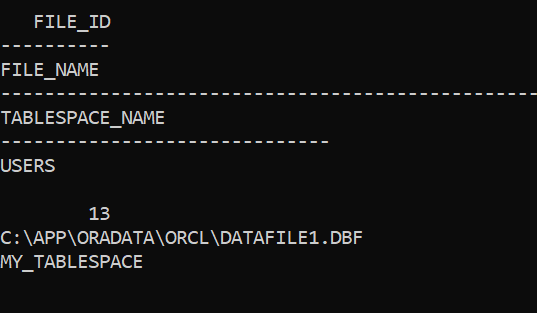
When we create a tablespace, you specify one or more datafiles that will belong to that tablespace. These datafiles determine the physical location where the data associated with the objects in the tablespace will be stored.

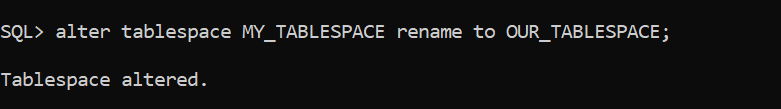
a.create tablespace

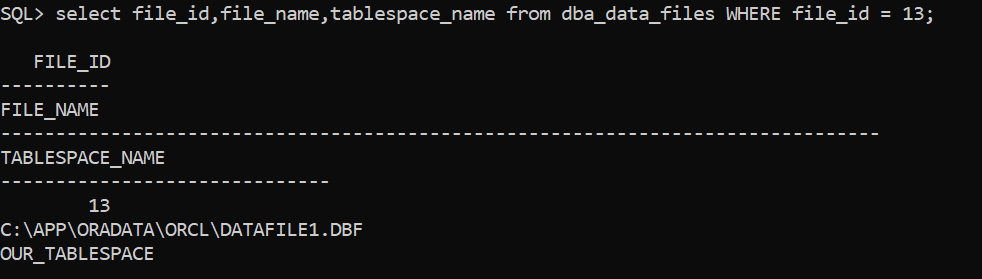


b.rename tablespace

Alter tablespace MY\_TABLESPACE rename to OUR\_TABLESPACE;







c. add data file

d.rename datafile

e.datafile resize

f.drop datafile

g.drop tablespace

2.Write steps to bring the database in archivelog mode

3.Write steps to multiplex the archive redologs.

**Lab Assignment#5**

Backup and Recovery:

1. Write a command to take a RMAN backup of all database in CDB.
2. Write a command to take a backup of specific pluggable database.
3. Write a command to take partial backup at tablespace and datafile level.
4. Write a command to restore and recover the database in following scenario:
   1. All Datafile loss
   2. One of the user defined datafile loss
   3. Control file loss
   4. Parameter file loss
5. Write a command to take a logical backup and recovery using datapump utility at following level:
   1. Full backup and recovery
   2. Table backup and recovery
   3. Schema level backup and recovery